Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 179 (2007) 234–252
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Computing, Artificial Intelligence and Information Management

A methodology for developing Bayesian networks: An application to information technology (IT) implementation

Eitel J.M. Lauría [a,*], Peter J. Duchessi [b,1]

[a] School of Computer Science and Mathematics, Marist College, 3399 North Road, Poughkeepsie, NY 12601, USA
[b] School of Business, University at Albany, State University of New York, 1400 Washington Avenue, Albany, NY 12222, USA

Received 13 July 2005; accepted 17 January 2006
Available online 10 March 2006

Abstract

Bayesian Networks (BNs) are probabilistic inference engines that support reasoning under uncertainty. This article presents a methodology for building an information technology (IT) implementation BN from client–server survey data. The article also demonstrates how to use the BN to predict the attainment of IT benefits, given specific implementation characteristics (e.g., application complexity) and activities (e.g., reengineering). The BN is an outcome of a machine learning process that finds the network's structure and its associated parameters, which best fit the data. The article will be of interest to academicians who want to learn more about building BNs from real data and practitioners who are interested in IT implementation models that make probabilistic statements about certain implementation decisions.

Keywords: Decision support systems; Bayesian networks; IT implementation; Artificial intelligence

1. Introduction

Companies rely increasingly on information technology (IT) to maintain their competitiveness. However, the IT implementation track record is not (and has not been) very good. Keil et al. (2000) suggest that at least one in four projects end in failure and 30–40% of all projects are less successful than expected. It is no wonder that IT implementation is the most researched topic in the IT community. The reasons for such negative results fall into varied categories, including organizational arrangements for IT implementation, organizational support, and project management. Benjamin and Levinson (1993) report that companies attain IT benefits because they do an especially good job at managing business process, organizational structure, and culture change issues rather than technological change issues. According to Griffith and Northcraft (1996), less than 10% of technology implementation failures result from technical problems; human and/or organizational issues account for most failures.

In general, to develop IT implementation models, information systems researchers have applied structural equation modeling (SEM) techniques (e.g., LISREL). In fact, the IT implementation literature

* Corresponding author. Tel.: +1 845 575 3000x2598.
E-mail addresses: Eitel.Lauria@marist.edu (E.J.M. Lauría), p.duchessi@albany.edu (P.J. Duchessi).
[1] Tel.: +1 518 442 4945.

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2006.01.016

is replete with research that incorporates SEM techniques. This paper takes a different approach and introduces another technique, a Bayesian Network (BN), for building an IT implementation model. A BN can serve as a probabilistic inference engine and provide unique perspectives about implementation activities, steps taken, and consequent benefits. The paper's primary objectives are: (1) provide a short overview of IT implementation research and the techniques generally used for modeling purposes; (2) introduce a methodology for building a BN for IT implementation; (3) demonstrate the application of the methodology to a real data set; and (4) describe how one BN can be used to make probabilistic statements about IT implementation, specifically client–server implementation.

In satisfying the above objectives, the paper makes three important contributions: (1) introduces an alternate modeling technique for IT researchers to consider in their research; (2) provides a methodological framework for building an IT implementation Bayesian network and demonstrates its application; and (3) shares the trials and tribulations (e.g., model search) of the modeling process with prospective readers.[2] The methodology is noteworthy because it describes how to build a BN from sample data. Although the methodology and its application are the primary contributions here, the resultant BN is also notable because it is novel and effective, as demonstrated by high levels of predictive accuracy. To our knowledge, this is the first instance of an IT implementation BN. However, others have developed BNs for software process modeling and software project risk management, see Bibi and Stamelos (2004) and Chin-Feng Fan and Yuan-Chang Yu (2004), respectively. Moreover, Fenton et al. (2002) and Fenton et al. (2004) have employed BNs to better understand resource allocation decisions for software projects; and Stamelos et al. (2003) have used BNs for predicting software productivity.

First the paper provides a brief literature review of IT implementation to acquaint readers with some of the factors that impact IT implementations, and it compares the predominant techniques used to examine IT implementations, namely regression analysis, LISREL, and Partial Least Squares (PLS). Then the paper provides a brief primer on BNs. Next the paper describes the methodology, providing insight into data preparation, selection of pertinent algorithms, and final model selection, and it applies the methodology to a real data set to develop a BN for client–server implementation. Finally, the paper concludes with a discussion of the tradeoffs associated with building BNs and makes several summary remarks.

[2] We presented the methodology at a recent conference and were amazed at the number of questions about how to build BNs from sample data. This response induced us to write the present paper.

2. IT implementation

IT implementation has been an important research issue since the late 70s and there is considerable interest in determining the primary factors—and their interrelationships—that determine IT implementation success (or conversely failure) because companies stand to lose considerable sums of money, if their implementations are not successful (i.e., fail to achieve the intended benefits). We cite some of the available research, factors (especially those that are relevant here), and benefits, recognizing that we are just scratching the surface and apologize for omitting any notable papers in the process.

2.1. Some critical implementation factors

Ettlie et al. (1984), Cheney et al. (1986), and Brancheau et al. (1989) indicate that senior management support is essential for effective IS planning and IS implementation execution. Ravichandran and Rai (2000) identify top management leadership, a sophisticated management infrastructure, process management efficacy, and stakeholder participation, as important elements of a quality-oriented software development effort. Reich and Benbasat (2000) indicate that congruence of IT vision (i.e., long-term alignment), shared domain knowledge, and effective planning processes among business and IT executives contribute to IT implementation success. Dolan (1995) indicates that functional management should play an important role in any IT implementation and should be responsible for recommending new applications. Robey and Farrow (1982), and Baronas and Louis (1988) indicate that user involvement results in more useful and accessible information systems. Hamilton and Chervany (1981) suggest that an adequate interaction between IS members and end users increases the chance of success and the attainment of benefits. Kiernan (1995) suggests that, if employee involvement is not viewed

as part of the overall transformation, the sole addition of information technology is likely to lead to job dissatisfaction. Duchessi et al. (1989) indicate that certain organizational arrangements, namely a steering committee and project team, are a necessity for successful IS implementation. McKeen (1983) emphasizes the importance of meeting budget constraints and achieving target dates, and Cox et al. (1981) recommends the development of standards, policies, and procedures to promote a successful IT implementation. Berg (1992) suggests that companies should keep the number of third-party vendors at a minimum, and Schultheis and Bock (1994) recommend standardizing on as few vendors as possible.

For specific applications, implementation studies demonstrate the importance of many of these factors. Chwelos et al. (2001) suggest that readiness, perceived benefits, and environmental pressure are important determinants of EDI adoption, and an empirical study of client–server systems conducted by the IBM Consulting Group (1994) indicates that organizational readiness and application complexity are important implementation factors. Wixom and Watson (2001) via a data warehousing, PLS-derived model demonstrate that system quality positively affect perceived net benefits; management support and resources resolve organizational issues that arise during data warehouse implementations; and resources, user participation, and highly-skilled project team members increase the likelihood that warehousing projects will finish on-time, on-budget, and with the right functionality. Wright and Wright (2001) analyze Enterprise Resource Planning (ERP) systems implementations; they conclude that the implementation process impacts system reliability and that common implementation problems, including improperly trained personnel an inadequate reengineering efforts, increase implementation risk.

2.2. Techniques for IT implementation research

Researchers have relied on several multivariate statistical techniques to analyze IT implementation data. The techniques include factor analysis, multiple analysis of variance, and canonical correlation analysis. Researchers have also relied on three predominate modeling techniques: multivariate regression analysis and the SEM techniques, namely LISREL and PLS. A brief description of each technique and Bayesian networks, which originated from the artificial intelligence community, appears below.

Multiple regression analysis uses several independent variables to predict one dependent variable (see Table 1). Multiple regression analysis maximizes the

Table 1
Comparison between regression analysis, LISREL, partial least squares, and Bayesian networks

Categories	Regression analysis	LISREL	Partial least squares	Bayesian networks
Research objectives	Variance explanation	Theory confirmation	Causal explanation and prediction	Casual explanation and statistical inference
Functional form	Linear	Linear	Linear	Nonlinear
Distribution assumptions	Gaussian	Gaussian	Gaussian	No restrictions but typically multinomial
Sample size	Small–large	Medium–large	Small–large	Small–large
Theory dependency	Models based on theory	Models based on theory	Models based on theory	No restrictions: theory or data driven
Model appropriateness	Statistical significance of model	Statistical significance of models	Statistical significance of model	Performance on scoring functions, conditional independence test, and cross-validation tests
Underlying approach	Minimize unexplained variation	Minimize covariation	Minimize unexplained variation	Graphical models and statistical machine learning in artificial intelligence
Learning curve	Steep	Moderate	Moderate	Flat
Availability of tools	Widely available	Available	Available	Available

explained variation between actual and predicted values of the dependent variable. Researchers can incorporate interaction effects and high order polynomial terms to model curvilinear relationships. Multiple regression analysis assumes a Gaussian distribution for error terms and homogeneity of variance for each independent variable.

SEM techniques combine factor analysis and path analysis. They are based on hypothesized causal relationships among the variables and graphically depict the nature and strength of those relationships. Constructs, or concepts that cannot be observed directly, serve as dependent and independent variables. LISREL focuses on maximizing the explained covariation among the various constructs, while PLS, like multiple regression analysis, maximizes the explained variation among the various constructs. Generally, researchers use SEM techniques to model IT implementation, as measured by the increasing number of publications based on those approaches. Chin (1998) suggests that the popularity of these techniques stems from the proliferation of software packages that perform LISREL and PLS (e.g., PLS-PC) analyses. According to Britt (1997), SEM models facilitate dialogue about what constructs are important (or unimportant) and how they may be related to each another. All of the statistical models enhance our understanding of a phenomenon along several dimensions, including description, interpretation, explanation, and/or prediction, and are used to perform confirmatory, mid-range, and exploratory research.

BNs are graphical models based on the notion of conditional independence that subsume a wide range of statistical models including regression models, factor analysis models, and recursive structural equation models—overcoming the linear and asymptotic theory assumptions of these approaches. In fact, according to Scheines (1995), the likelihood surface for structural equation models (SEMs) with latent variables is often multi-modal. BNs provide an efficient means for building models of domains with intrinsic uncertainty. Additionally, it is their ability to efficiently make probabilistic statements that turns them into one potential tool for IT implementation modeling.

Similar to SEM models, BNs graphically portray the nature and strength of relationships—not necessarily hypothesized—among several constructs or variables. In a BN, a directed acyclic graph (DAG) represents a set of conditional independence constraints, or assumptions, among a given number of variables and their related conditional probability distributions. The process of developing BNs is divided in two major parts: (1) learning the relationships between variables, and (2) parameterizing the associated conditional distributions. Instead of formulating a network, or model, on the basis of theory and then testing it, as with the SEM techniques, BN techniques identify the network that fits the data best. Researchers can incorporate hypothesized relationships among the variables to constrain the search space and ensure face validity, as subsequently described.

Finally, as with SEM models, BNs can also be interpreted causally. However, the notion of causality is open to interpretation, has been the subject of much philosophical debate over the years, and—as a result—is controversial. Generally, researchers use statistical techniques to "demonstrate" causality. According to Lauritzen (2000), graphical models, in particular those based on DAGs, have natural causal interpretations and thus form a language in which causal concepts can be discussed and analyzed in precise terms. For a thorough comparison between SEM and BNs applied to causal modeling, see Anderson and Vastag (2004).

In a decision-making context, BNs are a more natural choice than SEM-based models. BNs can estimate the values of all variables in a network, while structural equation models limit estimation to just dependent variables. In fact, for a given network, by applying the rules of Bayesian inference, BNs can propagate the impact of changing one or more variable values (i.e., new evidence) on one or more of the remaining variables that comprise the network, estimating those variables' values and providing the associated probabilities. BNs can be thought of as probabilistic inference engines that can answer queries, or perform "what-if" analyses, about the variables in a network. BNs can provide diagnostic reasoning (i.e., reasoning "upwards" from effects to cause), predictive reasoning (i.e., reasoning "downwards" from cause to effect), intercausal reasoning (e.g., given two mutually exclusive causes, evidence on one of them "explains away" the other one) and classification (i.e., "training" a BN as a classifier to predict one of the admissible values of a dependent discrete variable). For example, in the case of diagnosis, given evidence of an effect, a BN provides the probabilities of related causes. In the case of prediction, given related causes, a BN can infer the probabilities of subsequent effects. SEM-based models are incapable of

this type of reasoning. As a result, BNs are extremely valuable for providing actionable information and advice, while simultaneously incorporating uncertainty via their probability distributions. Generally, researchers use SEM techniques for just validating instruments, validating model constructs, and testing relationships between constructs. Thus, researchers use SEM techniques for explanation rather than estimation of variable values.[3]

3. A primer on Bayesian networks

BNs are graphical models that combine elements of both graph and probability theory. Broadly stated, BNs are directed acyclic graphs (DAGs) with a set of probability tables. A BN encodes the probability distribution of a set of random variables by specifying a set of conditional independence assumptions together with a set of relationships among these variables and their related joint probabilities. DAGs model the set of relationships among the variables; each of a DAG's nodes represents a variable and the arcs are the causal or influential links between the variables. A set of conditional probability functions associated with each node model the uncertain relationship between the variable and its parents. As pointed by Russell and Norvig (2003), BNs' conditional independence assumptions yield more compact models than those based on full joint probability distributions, relaxing the issue of computational complexity when considering a large number of variables. This visual layout provides a powerful knowledge representation formalism.

3.1. Structure and parameters of Bayesian networks

A DAG, say $\mathscr{G}$, depicts a set of variables, $X_1, X_2, \ldots, X_n$, and their relationships. Each node of the DAG $\mathscr{G}$ is a variable and the directed arcs represent the parental relationships among the variables (see Fig. 1). At each node, there is a conditional probability distribution for the variable given its parents' values. The DAG's joint probability distribution is

$$p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n|\mathscr{G}) = p(x_1, x_2, \ldots, x_n|\mathscr{G}) = \prod_{i=1}^{n} p(x_i \mid x_{\mathrm{pa}(i)}, \mathscr{G}). \quad (1)$$

In this notation, $p(x_1, x_2, \ldots, x_n|\mathscr{G})$ represents the probability of a specific combination of $x_1, x_2, \ldots, x_n$ from the variables $X_1, X_2, \ldots, X_n$, and $x_{\mathrm{pa}(i)}$ represents, as a vector, the list of direct parents of X_i, as depicted by DAG $\mathscr{G}$. BNs are locally structured, meaning that each node interacts with just its parent nodes. For discrete random variables, the conditional probability distributions are a set of tables, expressing a multinomial distribution, with parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_n\} = \{[\theta_{ik}(j)]_{k=1}^{r_i}\}_{j=1}^{q_i}$, where $i = 1 \ldots n$ identifies each variable $X_i \in \mathbf{X}$; $k = 1 \ldots r_i$ identifies each of the values of X_i; $j = 1 \ldots q_i$ identifies the set of valid configurations of values of the parent variables of X_i ($x_{\mathrm{pa}(i)}$). Restating Eq. (1) for discrete distributions, the DAG's joint probability distribution is:

$$p(\mathbf{x}|\boldsymbol{\theta}, \mathscr{G}) = \prod_{i=1}^{n} p(x_i|x_{\mathrm{pa}(i)}, \boldsymbol{\theta}_i, \mathscr{G}) = \prod_{i=1}^{n} \theta_{ik}(j). \quad (2)$$

3.2. Learning Bayesian networks from data

There are two possible scenarios to consider when learning a BN from a sample of N cases, $D = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$, where each $\mathbf{x}^{(k)}$ is a row vector representing a conjunction of observed states $x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}$ of $X_1, X_2, \ldots, X_n$: (1) the structure of the BN is known and parameters must be estimated; and (2) the structure of the BN is not known and, as a result, it must be found and the parameters estimated.

Concerning the former scenario, if the data is fully observable, modelers can apply maximum a posteriori (MAP) estimation to compute a BN's parameters. For discrete BNs, MAP estimates are calculated by counting the relative frequencies of occurrence of the values of each variable given each configuration of parent values. MAP estimation requires a parameter prior probability; generally, modelers use a Dirichlet parameter prior distribution, conjugate to the likelihood function's multinomial distribution. Appendix A provides a more detailed account of MAP estimation.

In the latter scenario, modelers have to determine the underlying structure of the BN given by DAG $\mathscr{G}$, which includes the specification of the conditional independence assumptions among the

[3] It is not our intention to suggest that one technique is better than another, but just to point out some prominent differences that may be important in a decision-making context.

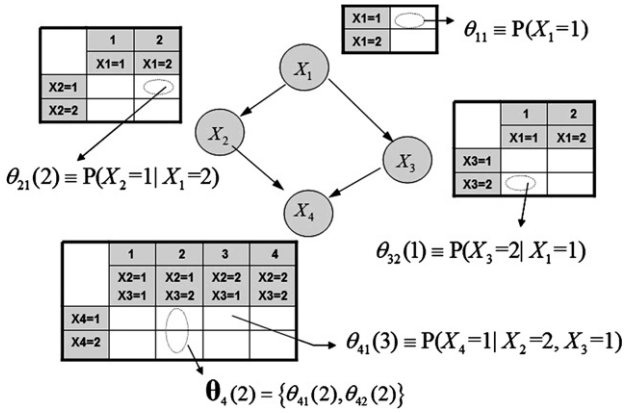

Fig. 1. Conditional probability distributions of a BN.

variables of the model, and the parameters θ. The literature describes two general approaches applied to the task of learning network structure from data: constraint-based methods, see Pearl and Verma (1991) and Spirtes et al. (2000), and search-and-score-based methods. Constraint-based methods are efficient, but they can lack robustness in practice: an algorithm's outcome is extremely sensitive to failures of conditional independency statistical tests. Thus, in most cases, researchers use search-and-score methods. Heuristic search techniques are available to find the best $\mathscr{G}$ and associated probability distributions over the space of all possible BNs. Greedy search algorithms, such as K2, see Cooper and Herskovits (1992), and Markov Chain Monte Carlo methods, such as simulated annealing, see Kirkpatrick et al. (1983), heuristically search the space of BN structures, using a scoring function to choose among the candidate networks. Several scoring functions can be considered, including Bayesian scores based on the posterior probability of the DAG $\mathscr{G}$, see Cooper and Herskovits (1992), or approximations of the posterior probability based on the Bayesian Information Criteria, see Schwarz (1978). Lam and Bachus (1994) propose a score based on the Minimum Description Length (MDL) Principle, see Rissanen (1986). Rodríguez (2001) and Lauría (2005) have applied an information-geometric (Info-geo) score that takes into account the volume of the manifold that maps the underlying statistical model, see Rissanen (1996) and Amari (1985). For a detailed description of this approach, see Lauría (2005).

As there may be many DAGs that determine the same joint probability distribution, the family of all DAGs with a given set of vertices is naturally partitioned into Markov-equivalence classes, each class being associated with a unique statistical model. This means that the structure of the network can only be learned up to its Markov equivalent class.

3.3. Inferencing in Bayesian networks

Because a BN defines a joint probability distribution, a fully parameterized BN can compute any probability from the network by specifying the joint probability distribution and applying basic rules of probability, such as the product rule and marginalization. Inferential tasks include diagnosis, prediction, inter-causal reasoning, and classification. Pearl (1988) and Lauritzen and Spiegelhalter (1988) provide several exact inference algorithms (e.g., message passing and junction tree). Neal (1993), Heckerman et al. (1994), and Jordan (1998) provide approximate inference algorithms for intractably large BNs. For more details on BN inferencing see Cowell et al. (1999) and Korb and Nicholson (2004).

4. Methodology

The methodology for building an IT Implementation BN consists of six sequential steps, namely Reduce Dimensionality; Rescale and Discretize Variables; Define Semantic Constraints; Search for Candidate Models; Benchmark Candidate Models; and Apply Selected Model. The first five steps focus on building a network that best fits the data, while the last step applies the network to a particular problem.

Step 1, Reduce Dimensionality, advocates using a dimensionality reduction technique to reduce the

number of variables and parameters requiring estimation. In general, as the number of variables and parameters grow, the number of cases required to estimate those variables and parameters grow exponentially. This problem is sometimes referred to as the "curse" of dimensionality, see Bellman (1961). Thus, for a given sample size, there are a maximum number of variables and parameters that can be estimated with accuracy. In practice, this problem is particularly acute for networks that are based on discrete, or dimensional (i.e., many variables having multiple values), data because there are many more possible combinations of variable values than can be supported with typical data sets—even with large data mining data sets. Several techniques are available to reduce dimensionality, including factor analysis; principal component analysis (PCA); multi-dimensional scaling (MDS); and more recently nonlinear manifold learning techniques, such as local linear embedding, see Roweis and Saul (2000), and Isomap embedding, see Tenenbaum et al. (2000). PCA, Factor Analysis, and MDS are eigenvector methods. PCA computes the linear projections of greatest variance from the top eigenvectors of the covariance matrix. Factor Analysis represents the shared variance of variables, excluding unique variance, and therefore models the correlation structure of the data. PCA is often used as a method for data reduction, while factor analysis is often preferred when the goal of the analysis is to detect structure. MDS computes the low dimensional projections that best preserve pair wise distances between data points. Nonlinear manifold learning techniques are appropriate for certain types of high dimensional data (e.g., human face recognition data sets) that may constitute highly nonlinear manifolds, which contain nonlinear structures that are invisible to PCA, factor analysis, or MDS.

In Step 2, Rescale and Discretize Variables, modelers may rescale continuous distributions and/or discretize continuous variables, according to some defined scheme, because the most commonly used network development algorithms require discrete rather than continuous variables. For awkward (e.g., highly skewed) continuous distributions, modelers can rescale the distributions using logarithmic or square root transformations; however, they often lose scale interpretability as a result. Concerning discretization, modelers can use various discretization schemes and binning criteria (e.g., equal width or equal frequency intervals) to create reasonable intervals.

The size of the data set may constrain the number of bins, or intervals, used, as it does the number of variables and parameters estimated. (Modelers need to ensure that each bin is represented in the data set.) Also, paring the number of intervals reduces the number of parameters that can be accurately estimated. For example, in a 14 variable binary network, where each variable can have only binary (i.e., two values), say "low" and "high," if each node has four parents, on average, there would be 244 (14×2^4) parameters to estimate; if each variable has three values, say "low," "medium," and "high," the number of parameters would increase to 1134 (14×3^4).

Model builders should realize that data reduction techniques and the use of nominal, low-level scales (e.g., the binary scale of "low" and "high") will reduce the granularity of the requisite measures and limit the accuracy of inputs, outputs, and associated probabilities. The granularity of any analysis is a function of the number of available cases: the greater the number of cases the greater the granularity of the analysis permitted. Consequently, model builders should strive for large data sets, which would allow them to go beyond a nominal level of measurement and incorporate higher measurement levels, including ordinal, interval, and ratio scales, which permit greater accuracy in measuring inputs and outputs. Conventional network development algorithms work best when there are no missing values (i.e., no intervals with missing values). For example, when there are no missing values, modelers can use maximum likelihood estimation rather than the more expensive expectation-maximization algorithm at each network search step.

Step 3, Define Semantic Constraints, recommends incorporating context-specific semantic constraints into the procedure that searches for the best network. This recommendation applies to any search procedure from exhaustive to less-exhaustive heuristic search procedures. An exhaustive search is not viable in practice, given the fact that the space of DAGs grows exponentially with the number of variables in the model. For example with three variables, the space of network structures is limited to 25 DAGs (11 Markov equivalent classes), whereas with 10 variables, this number skyrockets to 3×10^{17} DAGs (1×10^{17} Markov equivalent classes). Semantic constraints reduce the search space to only those networks that comply with time precedence or other dependency requirements among the variables, which the constraints stipulate. A smaller

search space results in less processing time to complete the search. Semantic constraints allow modelers to (1) incorporate previous information and knowledge that affect the configuration of a network, and (2) develop models that have face validity (i.e., have IT implementation activities as antecedents to IT benefits rather than the reverse). For example, there is a considerable body of literature that suggests that formal project management precedes and positively affects the attainment if IT implementation benefits. Basic theories about causal structure among variables and expert opinions are good sources of semantic constraints. At a minimum, modelers can assert, with enough accuracy, time precedence relationships among variables without introducing significant bias into the network search process.

Some researchers suggest that the use of semantic constraints grows in importance for large variable sets, or large-scale problems. The constraints limit the number of feasible BNs examined during the search process and increase the chance of producing a sensible model. In the case of large-scale problems, by applying domain knowledge, some researchers have demonstrated dramatic reductions in the number of node combinations considered without forsaking their models' ability to make sensible estimations (Neil et al., 2000).

Step 4, Search for Candidate Models, applies search-and-score-based, heuristic search algorithms to search the space of networks, identify several high potential candidate networks, and estimate their associated parameters. Each combination of scoring function, search algorithm, and equivalent sample size will produce a candidate network. The process of searching for a suitable network can be as much and art as it is a science, as are other modeling exercises that use conventional techniques (e.g., PLS). The scoring functions and search algorithms make learning network structure an optimization problem, where the goal is to find the network, or DAG $\mathscr{G}$ (or any member of its Markov equivalent class), in the space of network structures $\mathbb{S}^{\mathscr{G}}$ that maximizes a score and represents the best fit with the training data set D.

A natural scoring function is the network's posterior probability distribution $p(\mathscr{G}|D) \propto p(\mathscr{G}|D) \times p(\mathscr{G})$. Generally, it is impractical to compute the exact value of $p(\mathscr{G}|D)$, because even for small networks the computation becomes intractable. Consequently, modelers can use simplifying assumptions (e.g., on the nature of the distributions) to estimate $p(\mathscr{G}|D)$. The K2 algorithm, see Cooper and Herskovits (1992), adopts a uniform structure prior $p(\mathscr{G})$ and computes the marginal likelihood $p(D|\mathscr{G})$ assuming conjugate Dirichlet parameter priors. K2 chooses an order over the variables of the BN and uses a greedy local search procedure that, for each variable X_i in the ordering, incrementally adds as a parent that predecessor $(X_1, X_2, \ldots, X_{i-1})$ whose addition most increases the score. This procedure is repeated independently for each variable X_i until no node increases the score, or the number of parents of X_i exceeds a given threshold. (Note that the local search algorithm implemented by K2 can lead to just local maxima).

Alternatively, in lieu of exact results, other scoring functions can provide simple approximations of the network's posterior probability distribution. The Bayesian Information Criteria (BIC) score is a large sample estimate of the marginal likelihood of the model, but in practice the sample size does not have to be large to get a good approximation. Besides, it does not require a parameter prior.

The Info-geo scoring function refines the BIC score by incorporating an additional term to BIC's formulation, as it is based on the MDL Principle, see Rissanen (1996) and Lauría (2005). (The info-geo scoring function takes the form $-\log p(D|\hat{\boldsymbol{\theta}}) + \frac{|\boldsymbol{\theta}|}{2}\log\frac{N}{2\pi} + \log\int\sqrt{\det \mathbf{I}(\boldsymbol{\theta})}\,\mathrm{d}\boldsymbol{\theta}$, where the first term is the logarithm of the likelihood at the maximum likelihood estimator $\hat{\boldsymbol{\theta}}$, the second term measures the complexity given by the number of parameters in the model, and the last term, which includes the determinant of the Fisher Information matrix $\mathbf{I}(\boldsymbol{\theta})$, measures the 'geometrical' complexity given by the volume of the BN's manifold.[4] Note that the first two terms correspond to the BIC score with a negative sign.)

To search for global optimal scores, modelers can apply genetic search algorithms, see Holland (1975), or MCMC search algorithms, such as simulated annealing, see Kirkpatrick et al. (1983). For example, simulated annealing considers each network a state of a Markov Chain. At each step of the search procedure, the algorithm perturbs a network to jump from one state of the Markov chain to another. Perturbations are three operations: arc addition, arc removal or arc reversal. The algorithm

[4] The expression $\int\sqrt{\det \mathbf{I}(\boldsymbol{\theta})}\,\mathrm{d}\boldsymbol{\theta}$ corresponds to the volume of the Riemannian manifold on which the parametric family of probability distributions of the BN is mapped.

creates a list of potential neighbor networks to explore by applying the three operations and randomly picks one network from the list to score with a given scoring function. The algorithm selects networks with improved scores for further processing, while it rejects networks with decreasing scores with a finite probability. The algorithm constrains the space of networks examined via acyclicity checks and the semantic constraints of the restriction matrix. This approach may take more time than a greedy search algorithm, but it has a high probability of converging to the global maximum (the algorithm will find the global maximum given enough time). For a detailed description of the simulated annealing algorithm see Appendix B.

In Step 5, Benchmark Candidate Models, modelers compare the candidate networks that emerge from Step 4, to identify the network that provides the best representation of the domain or process being modeled. Several evaluation methods can be considered. Predictive accuracy is far and away the most popular evaluation method to assess probabilistic models, including Bayesian networks. In practice, for a given fixed data set, modelers generally use tenfold cross-validation, see Witten and Frank (2000), to determine the predictive accuracy and thus validate the network. Cost-sensitive classification methods that compute the best weighted average cost or benefit from their probabilistic predictions, can also be applied so long as the costs associated with misclassification and the benefits of correct classification are available, see Korb and Nicholson (2004). The Kullback–Leibler distance is an idealized metric that compares the true joint probability distribution—which is never known—with the learned joint probability distribution. Consequently, it can be considered as just a reference standard.

Finally, in Step 6, Apply Selected Model, modelers apply the network with the highest performance evaluation metric to make probabilistic statements about the domain or process being modeled. At this point, modelers can use the network as an inference engine, as demonstrated shortly.

5. Demonstration data set

The data for deriving one IT implementation BN is based on a nation-wide survey of companies implementing client–server systems, see Chengalur-Smith and Duchessi (2000). The study identifies the primary factors that affect successful client–server implementations and explicates associated benefits and potential pitfalls. The data set consists of 350 respondents, who completed a questionnaire containing a set of structured, fixed response questions. Most of the questions, except those that require a specific input (e.g., project budget), use 5-point Likert scales or rating scales with defined anchors. The majority of respondents to the survey are senior IT executives (41%); 24% senior corporate executives (including CEOs); and the rest (35%) primarily IS and functional managers. Almost 75% of the respondents are from companies that have implemented a complex client–server application. Just over 60% of the respondents' businesses are in the service (25%), industrial (18%), and financial (18%) sectors. About 60% of the applications span three or more functional areas, and 75% of the respondents describe their applications as mission critical. The original data set consists of a wide variety of client–server applications, a few continuous variables, and a large number of discrete (e.g., Likert and rating scale) variables. We use exclusively the discrete variables.

6. Application of the methodology

Given the above data set, the following describes one application of the methodology, the network development algorithms used, and the resulting client–server implementation BN, or model.

6.1. Step 1: Reduce dimensionality

Due to the limited number of cases, to develop a more parsimonious set of variables, we use factor analysis to identify the important client–server implementation constructs. The resulting factors are the primary variables of the network. The factor analysis identifies 13 implementation and four benefit factors. IS Maintenance represents the degree to which the IS group maintains client–server applications. Project Management portrays the quality of the roll-out and ability to meet target dates. Vendor Support depicts the extent of vendor support. Business Planning represents the quality of business planning and IT strategy formulation. Business Planning includes defining business goals and strategies; integrating business and IT strategies; and having the CIO make strong contributions to strategy formulation. IS/User Interaction evinces the working relationship between the IS Group and users. IS/User Interaction includes having the IS

groups work with users as partners and use formal methods of evaluating vendors. Organizational Arrangements represents the earnestness of the steering committee and quality of project team to provide adequate resources and implementation direction. IS Management's Role represents IS management's ability to integrate business requirements into IT strategies, plan computing infrastructure, and align an application portfolio with goals and strategies. Functional Management's Role measures functional management's contribution to IT strategy and requirements analysis. Application Complexity represents the overall complexity (e.g., implementation involves all business functions) of a client–server implementation project. Application Suitability represents the ability of the client–server application to enhance users' performance. Reengineering Intention depicts a company's intention to redesign and eventual redesign of business processes during the implementation effort. Application Analysis represents the approach for soliciting users' requirements and testing system evaluation. Application Analysis includes allowing users to provide input into system design and using pilots and test systems during the implementation. Standards and Integration measures the quality of vendor evaluation and systems integration. Additionally, we reduce the four benefit factors (Operational, Competitive, System, and Financial Benefits), which include reduced decision-making time, improved integration of business information, and better control of the business, into one benefit factor, Benefits. Thus, there are 14 variables, or network nodes, for developing the client–server implementation network.

6.2. Step 2: Rescale and discretize variables

All 14 variables are continuous with values that range from one to five and, as a result, are easy to interpret. Consequently, we do not rescale the data. We discretize the data via a binary, or two-value, scheme, where 1 = "low" and 2 = "high." For example, Application Complexity is either "low" or "high" for simple, single-level, single-function client–server applications and for mission critical, multi-level, multi-function client–server applications, respectively. The size of the existing data set just does not permit greater granularity (e.g., where 1 = "low," 2 = "medium," and 3 = "high"). Assuming an average of four parents per variable, our binary scheme with fourteen variables requires the estimation of 224 (14×2^4) parameters, which compares favorably to the 312 available cases. (We lost 48 cases due to missing values.) In reality there are only 281 admissible cases for learning the network; the remaining 31 cases represent a hold-out sample for a forthcoming cross-validation analysis, see Step 5 below.

6.3. Step 3: Define semantic constraints

To incorporate existing IT implementation knowledge and our expertise and to ensure time precedence and face validity, we use semantic constraints. Collectively, the constraints enforce, in rather loose manner, the implementation, precedence, and validity conditions, while allowing for ample identification of relationships between the unconstrained variables, as the search algorithms examine alternate networks.

We use two algorithms, namely K2 and simulating annealing, to heuristically search the network space (see Step 4 below). For simulated annealing, we use a restriction matrix to portray the semantic constraints (see Table 2). The restriction matrix has the existing variables as both rows and columns. Each cell in the matrix represents a potential relationship, or DAG-directed edge, given by an ordered row-column pair. A cell containing a 1 defines a constraint, allowing a search algorithm to eliminate from consideration all networks with that relationship. For example, the restriction matrix enforces time precedence rules between Benefits and Business Planning via the 1 in the Benefits-Business Planning cell. Networks with relationships pointing from Benefits to Business Planning are invalid.

K2 requires ordered variables. For consistency purposes, we order the variables following the criteria given by the restriction matrix used in simulated annealing. We compute the column totals of the restriction matrix and use the totals to rank the nodes: those nodes with a smaller number of restrictions ranked higher in the ordering. For those nodes with equal number of restrictions, we apply face validity considerations and our own expertise. The resulting ordering of the variables is Business Planning, IS Management's Role, Functional Management's Role, Organizational Arrangements, Reengineering Intensions, Standards and Integration, Vendor Support, Application Complexity, Application Analysis, IS Maintenance, IS/User Interaction, Project Management, Application Suitability, and Benefits.

Table 2
Semantic restriction matrix used in simulated annealing

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		IS Mainten.	Project Mgmt	Vendor Support	Organization Arrangements	Business Planning	IS–User Interaction	IS Mgmt Role	Functional Mgmt Role	Application Complexity	Application Suitability	Reengineer Intentions	Application Analysis	Standards & Integration	Benefits
1	IS Mainten.	1	0	1	1	1	0	1	1	1	0	1	1	1	0
2	Project Mgmt.	1	1	1	1	1	1	1	1	1	0	1	1	1	0
3	Vendor Support	1	0	1	1	1	1	1	1	1	0	1	0	1	0
4	Org. Arrang	0	0	0	1	1	0	1	1	0	0	0	0	0	0
5	Business Planning	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6	IS–User Interaction	0	0	1	1	1	0	1	1	0	0	1	0	0	0
7	IS Mgmt Role	0	0	0	0	1	0	1	0	0	0	0	0	0	0
8	Functional Mgmt role	0	0	0	0	1	0	0	1	0	0	0	0	0	0
9	Application Complexity	0	0	1	0	1	0	1	1	1	0	1	0	1	0
10	Application Suitability	0	1	1	1	1	1	1	1	1	0	0	1	1	0
11	Reeng Intentions	1	0	0	0	1	0	1	1	1	0	1	0	0	0
12	Application Analysis	0	0	0	1	1	0	1	1	0	0	0	1	0	0
13	Stds & Integration	0	1	0	1	1	1	1	1	0	0	1	0	1	0
14	Benefits	1	1	1	1	1	1	1	1	1	1	1	1	1	1

6.4. Step 4: Search for candidate models

To explore the space of potential BNs and learn the candidate networks, we use several combinations of heuristic search algorithms (K2 and simulated annealing), scoring functions (BIC and Info-geo), and Dirichlet equivalent sample sizes (1, 5, and 10). Each combination employed produces a network. In Step 5 below, we select the final network: the one that provides the highest overall predictive accuracy.

K2 imposes a partial ordering on the network variables to constrain the search space, using a greedy algorithm to perform the search. Simulated annealing performs a global optimization and is not restricted by the ordering of the nodes, but it is typically slower than greedy search algorithms such as K2, and can take time to converge to a global optimum.

We apply two scoring functions, BIC and Info-geo, with simulated annealing. With the Info-geo scoring function, the algorithm incorporates an estimate of the volume of the parameter manifold, in order to minimize the impact of recalculating the score at each step of the search procedure. This feature requires modified versions of simulated annealing so as to include at each step the difference of volumes between successive networks. Appendix B provides a more detailed account of the search process.

6.5. Step 5: Benchmark candidate models

We use tenfold cross-validation to determine the predictive accuracy of the networks emerging from the immediately preceding step. We divide the data set into 10 partitions, or folds, containing 31 cases each. Each fold is held in abeyance, as a hold-out fold, and the search procedure operates on the remaining nine folds, or 281 cases. We use the hold-out fold to estimate the predictive error on all 14 variables. Thus, we execute the learning procedure 10 times and the validation procedure 10 times, using the different training sets. For each candidate network, we average the 140 error estimates (14 variables $\times$ 10 folds) to compute an overall error estimate for a network. We select the candidate network with the highest overall predictive accuracy as the final network, or model (see Table 3). The network structure and parameter estimates represent a fully specified BN for client–server implementation (see Figs. 2 and 3).

Table 3
Predictive accuracy of final model

	Node #														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Overall predictive accuracy
Predictive accuracy	91.90	78.41	72.73	80.42	81.82	79.63	91.14	97.92	73.84	83.05	79.22	83.86	79.22	77.77	82.21

Nodes: 1 = IS Maintenance; 2 = Project Mgmt.; 3 = Vendor Support; 4 = Organization Arrangements; 5 = Business Planning; 6 = IS–User Interaction; 7 = IS Mgmt. Role; 8 = Functional Mgmt. Role; 9 = Application Complexity; 10 = Application Suitability; 11 = Reengineering Intentions; 12 = Application Analysis; 13 = Standards & Integration; 14 = Benefits.

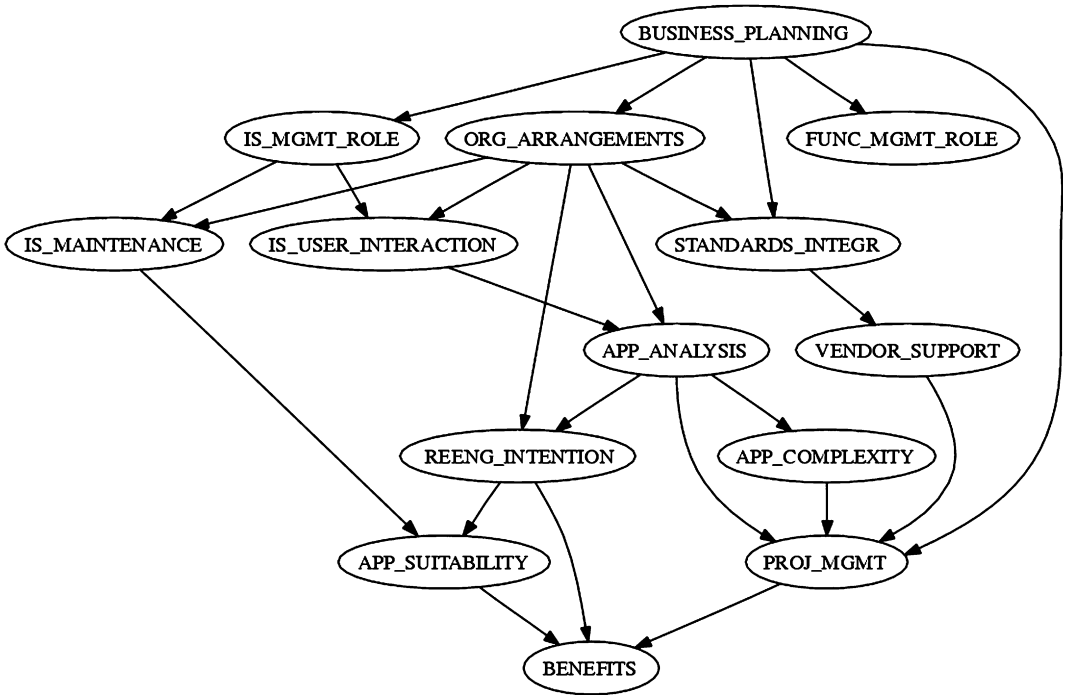

Fig. 2. Client–server implementation final model.

6.6. Step 6: Apply selected model

The empirically derived network shows important directed dependencies and conditional independencies between the 14 variables. For example, according to the network's DAG, Reengineering Intention (REENG_INTENTION), Application Suitability (APP_SUITABILITY), and Project Management (PROJ_MGMT) directly affect the attainment of Benefits (BENEFITS) in a client–server implementation (see Fig. 2 again). Moreover, through its parameter estimates, the network assesses the chance of attaining 1 = "low" and 2 = "high" Benefits given various variable values, namely 1 = "low" and 2 = "high." For example, the network demonstrates that well-planned projects (Node 2, PROJ_MGMT = 2) delivering suitable applications (Node 10, APP_SUITABILITY = 2) based on reengineered processes (Node 11, REEING_INTENTION = 2) provide a high level of Benefits (Node 14, BENEFITS = 2) 83.3% of the time (see Fig. 3 again). Conversely, the combination of a poorly planned project (Node 2, PROJ_MGMT = 1), an ill-fitting application (Node 10, APP_SUITABILITY = 1), and no process reengineering (Node 11, REENG_INTENTION = 1) imply a low level of Benefits (Node 14, BENEFITS = 1) 81.1% of the time. Even for poorly

Nodes — **parents = 4, 7**

		1	2	1	2
		1	1	2	2
1	1	0.862	0.804	0.546	0.261
	2	0.139	0.197	0.454	0.739

parents = 3, 5, 9, 12

		1	2	1	2	1	2	1	2
		1	1	2	2	1	1	2	2
		1	1	1	1	2	2	2	2
		1	1	1	1	1	1	1	1
	1	0.996	0.791	0.837	0.391	0.724	0.694	0.439	0.500
	2	0.004	0.209	0.163	0.609	0.276	0.306	0.561	0.500

2

		1	2	1	2	1	2	1	2
		1	1	2	2	1	1	2	2
		1	1	1	1	2	2	2	2
		2	2	2	2	2	2	2	2
	1	0.590	0.422	0.336	0.264	0.597	0.529	0.371	0.231
	2	0.410	0.578	0.664	0.736	0.403	0.471	0.629	0.769

parents = 13

		1	2
3	1	0.628	0.421
	2	0.372	0.579

parents = 5

		1	2
4	1	0.691	0.426
	2	0.309	0.574

5

1	0.554	
2	0.446	

parents = 4, 7

		1	2	1	2
		1	1	2	2
6	1	0.759	0.492	0.454	0.177
	2	0.241	0.508	0.546	0.823

parents = 5

		1	2
7	1	0.737	0.440
	2	0.264	0.560

parents = 5

		1	2
8	1	0.907	0.263
	2	0.093	0.737

Nodes — **parents =12**

		1	2
9	1	0.676	0.546
	2	0.324	0.454

parents = 1, 11

		1	2	1	2
		1	1	2	2
10	1	0.729	0.365	0.570	0.162
	2	0.271	0.635	0.430	0.838

parents = 4, 12

		1	2	1	2
		1	1	2	2
11	1	0.745	0.489	0.516	0.325
	2	0.255	0.511	0.484	0.675

parents = 4, 6

		1	2	1	2
		1	1	2	2
12	1	0.739	0.466	0.508	0.259
	2	0.261	0.534	0.492	0.741

parents = 4, 5

		1	2	1	2
		1	1	2	2
13	1	0.731	0.399	0.525	0.279
	2	0.269	0.601	0.475	0.722

parents = 2, 10, 11

		1	2	1	2	1	2	1	2
		1	1	2	2	1	1	2	2
		1	1	1	1	2	2	2	2
14	1	0.811	0.715	0.542	0.605	0.570	0.500	0.336	0.168
	2	0.189	0.285	0.458	0.395	0.430	0.500	0.664	0.833

Nodes: **1=IS Maintenance; 2=Project Mgmt; 3=Vendor Support; 4=Organization Arrangements; 5=Business Planning; 6=IS - User Interaction; 7=IS Mgmt Role; 8=Functional Mgmt Role; 9=Application Complexity; 10=Application Suitability; 11=Reengineering Intentions; 12=Application Analysis; 13=Standards & Integration; 14=Benefits**

States: **1=Low; 2= High**

Fig. 3. Parameter estimates of the final model.

planned projects (Node 2, PROJ_MGMT = 1), there is still a 66.4% chance of obtaining a high level of Benefits (Node 14, BENEFITS = 2), as long as the client–server application is suitable (Node 10, APP_SUITABILITY = 2) and enables a reengineered business process (Node 11, REENG_INTENTION = 2). The other implementation factors indirectly affect Benefits through these three factors. Consequently, the network provides intermediate results about Application Suitability, Reengineering Intention, and Project Management. For example, Application Suitability (Node 10, APP_SUITABILITY) depends directly on IS Maintenance (NODE 1, IS_MAINTENANCE) and on Reengineering Intention (Node 11, REENG_INTENTION). Client–server implementations that have the strong support of the IS group in providing the adequate computing infrastructure (NODE 1, IS_MAINTENANCE = 2) and are based on reengineered business needs (Node 11, REENG_INTENTION = 2) are highly likely (83.8%) to produce suitable applications (Node 10, APP_SUITABILITY = 2). Conversely, client–server implementations that have little IS involvement (Node 1, IS_MAINTENANCE = 1) and include little reengineering (Node 11, REENG_INTENTION = 1) are highly likely (72.9%) to produce unsuitable applications (Node 10, APP_SUITABILITY = 1). Finally, concerning Application Suitability, IS involvement (Node 1, IS_MAINTENANCE = 2) renders, in every case, a higher probability of delivering a suitable application (Node 10, APP_SUITABILITY = 2).

This ability to make probabilistic assertions is an important benefit of the network. It allows a company to (1) determine its chance of success (i.e., attainment of a high level of benefits) given its current situation, and (2) perform ''what-if'' analyses on implementation activities and implementation steps taken.

Concerning situation assessment, consider a company with the following attributes: poorly defined goals and strategies, misaligned business and IT strategies, and negligible input from the CIO during strategy formulation (Node 5, BUSINESS_PLANNING = 1 with probability 1); IS management fails to integrate business requirements into IT strategy, does not plan computing infrastructure, and fails to align application portfolio with existing goals and strategies (Node 7, IS_MGMT_ROLE = 1 with probability 1); steering committee rarely provides adequate resources, project team rarely includes people with technical expertise, and the steering committee and project team generally fail to provide significant direction during an implementation (Node 4, ORG_ARRANGEMENTS = 1 with probability 1); IS group hardly works with users, users do not consider IS group as partners, IS group rarely selects vendors based using a formal methodology (Node 6, IS_USER_INTERACTION = 1 with probability 1); and vendor products do not integrate often, users have little input into the design of IT applications, and pilots and test systems are rarely during an implementation (Node 12, APP_ANALYSIS = 1 with probability 1). Additionally, the company is about to implement a complex client–server application that incorporates existing legacy systems and spans several business functions (Node 9, APP_COMPLEXITY = 2 with probability 1). Given this scenario, what can the company expect in terms of being able to successfully apply business process reengineering; develop a suitable application that improves users' performance; have a well-planned and timely roll-out; and obtain anticipated benefits, including improved system accessibility, increased organizational performance, and improved financial posture?

By applying the rules of Bayesian inference to propagate the impact the above evidence, the present BN indicates that the company has only a 19% change of reengineering (REENG_INTENTION = 2), slightly less than 50% chance of achieving a suitable application (APP_SUITABILITY = 2), no chance of being on time and having a well-planned roll-out (PROJ-MGMT = 2), and a 36% chance of attaining benefits (BENEFITS = 2).

With regard to ''what-if'' analyses, prior to pursuing a client–server implementation, suppose the company decides to revise specific IT implementation practices. For example the company makes improvements in the way it organizes for the implementation, including having the steering committee and project team meet regularly (ORG_ARRANGEMENTS = 2 with probability 1). The company also decides to involve users in all phases of the client–server project and use a pilot before going live with the client–server system (APP_ANALYSIS = 2 with probability 1). Finally, the company emphasizes the need to use plug-compatible components (STANDARDS_INTEGR = 2 with probability 1). Can the company expect to improve its chance of success, including intermediate dimensions (e.g., Reengineering Intention) and

final Benefits? Concerning the intermediate dimensions, by again applying the rules of Bayesian inference to propagate the new evidence, the present BN indicates that the company would have a 61% change of successfully reengineering the affected processes (REENG_INTENTION = 2), slightly less than 53% chance of achieving a suitable application (APP_SUITABILITY = 2), 47% chance of being on time and having a well-planned roll-out (PROJ_MGMT = 2), and just over a 50% chance of attaining anticipated client–server benefits (BENEFITS = 2). The company can continue with the "what if" analyses by using other combinations of factors and attribute values to determine the impact on intermediate factors and on the final benefits.

7. Discussion

The practical methodology presented here explains how to use survey data to build an IT implementation BN that makes probabilistic assertions about an implementation. As demonstrated, modelers will have to make tradeoffs when working with real data sets.

Based on a data set's characteristics and a project's overall modeling objectives, modelers will have to make decisions that will affect precision. For example, to work within the confines of our data set, we appeal to factors, or constructs, (e.g., Reengineering Intention) and discretize those factors with a binary scale, forcing us into a less granular analysis versus one based on a larger scale. Although we sacrifice granularity, we develop an enhanced understanding of IT implementation based on high-level constructs (e.g., Application Complexity). Additionally, modelers will have to decide which algorithms are most appropriate for their situations, recognizing the tradeoff between speed and potential solution quality. Simulated annealing and K2 with various combinations of scoring metrics and Dirichlet prior distributions to learn several candidate networks, before selecting a final model based on predictive accuracy. Although several alternative models are possible, especially when researchers experiment with multiple techniques, there is considerable robustness here. For example, all of our candidate networks are very similar, having approximately the same topology, with no arc reversals among them; they differ only in the number of arcs. The maximum topology dissimilarity between the final model and the rest of the candidates is 21% (5 dissimilar arcs), with a mean of 18%, revealing a substantial convergence of the candidate networks with the final network, or model.

Modelers will have to decide whether to use expert judgement and/or well-founded theory alone to determine a network's structure, allowing the data to determine just the parameter estimates, or to use the data to determine both structure and parameter estimates. We use a hybrid approach to developing our network, using semantic constraints to incorporate theory and expertise, while letting the data determine the remaining structure and parameter estimates. The quantity and quality of expertise and the extant theory are important inputs to determining an appropriate course of action.

Finally, when building any IT artifact, or system, effectiveness is a critical concern. Modelers will have to consider the tradeoff between additional testing and quick implementation. We use tenfold cross-validation, a common approach, to determine the classification accuracy of the final model. The acid test would be to apply the system to new and/or ongoing implementation cases other than those in the data set and evaluate its performance. However, this would be extremely difficult to do, considering that implementation projects, especially for large applications, are lengthy affairs. In general, modelers appeal to cross-validation procedures and a short pilot period for effective testing without unnecessarily delaying the introduction of a system.

8. Summary and conclusions

BNs offer an alternative to developing models for IT implementation. The methodology presented here demonstrates how to build an IT implementation BN from survey data. It covers a number of important activities (e.g., scaling and discretizing data) and several important modeling techniques (e.g., simulated annealing) for building BNs. The final network portrays the relationships among a number of important IT implementation factors (e.g., Project Management) and benefits. Additionally, it provides a means for making probabilistic assertions about those factors and the attainment of benefits. Although IT researchers have historically used SEM techniques to build IT implementation models, BNs may grow in importance, as the IT research community gains an enhanced awareness of them and begins using them to study IT implementation and other phenomena. This paper makes a step in that direction.

Appendix A. Learning BNs' parameters through MAP estimation

For a BN with DAG $\mathcal{G}$, and a given a dataset D of N independent cases, modelers can apply maximum a posteriori (MAP) estimation to compute the BN's parameters, represented as vector $\boldsymbol{\theta}$, that maximize the posterior probability $p(\boldsymbol{\theta}|D,\mathcal{G})$. Note that $p(\boldsymbol{\theta}|D,\mathcal{G})$ can be expressed in terms of the likelihood of the data $p(\boldsymbol{\theta}|D,\mathcal{G})$ and the prior probability $p(\boldsymbol{\theta}|\mathcal{G})$ as

$$p(\boldsymbol{\theta}|D,\mathcal{G}) \propto p(D|\boldsymbol{\theta},\mathcal{G}) \times p(\boldsymbol{\theta}|\mathcal{G}). \tag{3}$$

Following Ramoni and Sebastiani (1999), the likelihood $p(D|\boldsymbol{\theta},\mathcal{G})$ factorizes into a product of values $\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\theta_{ik}(j)^{N_{ik}(j)}$, each of which is in turn a product of factors $\prod_{k=1}^{r_i}\theta_{ik}(j)^{N_{ik}(j)}$ that depend on the parent configuration, where $N_{ik}(j)$ is the number of times variable X_i takes the value k, given parent configuration j. Assuming parameter independence on $\boldsymbol{\theta}_i(j)$ and no missing data, the posterior distribution of the parameters given a data set D can be calculated as

$$p(\boldsymbol{\theta}|D,\mathcal{G}) = \prod_{i=1}^{n}\prod_{j=1}^{q_i} p(\boldsymbol{\theta}_i(j)|\mathcal{G}) \prod_{k=1}^{r_i}\theta_{ik}(j)^{N_{ik}(j)}. \tag{4}$$

Using a Dirichlet prior probability distribution with parameters $D_i(\alpha_{ij1},\alpha_{ij1},\ldots,\alpha_{ijr_i})$ to regularize the sample, the maximum posterior estimates are:

$$\tilde{\theta}_{ik}(j) = \frac{N_{ik}(j)+\alpha_{ijk}}{\sum_{k=1}^{r_i}N_{ik}(j)+\alpha_{ijk}}. \tag{5}$$

The Dirichlet prior can be interpreted in terms of counts of imaginary observations. The quantity $\alpha_{ij} = \sum_{k=1}^{r_i}\alpha_{ijk}$ is the local precision on $\boldsymbol{\theta}_i(j)$, and $\alpha_i = \sum_{j=1}^{q_i}\alpha_{ij}$ is the global precision on $\boldsymbol{\theta}_i$, the number of imaginary cases for all the conjunctions $\{x_i^{(t)} = k$ and parent configuration $x_{\mathrm{pa}(i)}^{(t)} = j\}$ of a given node X_i. For consistency purposes, assuming that all variables X_i have the same prior sample size, $\alpha = \alpha_i$, α_{ijk} can be calculated from α as $\alpha_{ijk} = \alpha/(q_i \times r_i)$, distributing α uniformly among the parameters of each conditional probability distribution at each node. The global precision parameter α is known as the equivalent sample size and represents the strength of the parameter prior.

Appendix B. Simulated annealing algorithm design considerations

Simulated annealing, see Kirkpatrick et al. (1983), is a Markov Chain Monte Carlo approach to optimization of multivariate functions. The term derives from the physical process of heating and slow cooling of a solid material (typically glass or metal) in order to toughen the material and reduce brittleness. In an annealing process a melted solid, initially at high temperature and disordered, is slowly cooled so that the system at any time is approximately in a given state of thermodynamic equilibrium. In terms of statistical physics, the Boltzman distribution describes the probability distribution of a state $\boldsymbol{\theta}$ with energy $E(\boldsymbol{\theta})$ for a system in a heat bath at temperature T:

$$\pi(\boldsymbol{\theta}) \propto \exp\left(-\frac{E(\boldsymbol{\theta})}{T}\right). \tag{6}$$

The cooling stage of the annealing process is simulated by taking a sequence of slowly decreasing temperatures $T_0, T_1, T_2, \ldots$ converging to 0. Running the Metropolis–Hastings algorithm (see below) with each value of the sequence $T_0, T_1, T_2, \ldots$, results in a sequence of annealing states $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots$ with decreasing energies $E(\boldsymbol{\theta}_0), E(\boldsymbol{\theta}_1), E(\boldsymbol{\theta}_2), \ldots$. In the limit, when the temperature approaches 0, the system evolves towards an equilibrium state with minimum energy. This can be seen as a combinatorial optimization process where a sequence of feasible solutions gradually approach an optimal solution (global minimum) using the energy equation for the thermodynamic system as the objective function.

The Metropolis–Hastings algorithm, see Metropolis et al. (1953), Hastings (1970), simulates the evolution of a thermodynamic system from one configuration to another. Given an initial state $\boldsymbol{\theta}_0$ of the system at energy $E(\boldsymbol{\theta}_0)$ and temperature T, the initial configuration is perturbed ($\boldsymbol{\theta}_0 \to \boldsymbol{\theta}^*$) keeping T constant, and the change in energy $\Delta E = E(\boldsymbol{\theta}^*) - E(\boldsymbol{\theta}_0)$ is calculated. If $\Delta E < 0$, the new configuration $\boldsymbol{\theta}^*$ with lower energy is accepted. Otherwise $\boldsymbol{\theta}^*$ is accepted with finite probability given by $\frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}_0)} = \exp(-\Delta E/T)$.

Simulated annealing converges asymptotically to the optimal solution. The temperature T and number of steps of each Metropolis–Hastings run at each state $\boldsymbol{\theta}_i$ control the optimization process.

This same combinatorial optimization approach is applied to search the space of Bayesian network DAGs. Each network represents a state of a Markov chain. At each step in the search procedure the algorithm perturbs the network to randomly jump to a neighboring DAG. The algorithm performs any of three operations (arc addition, arc

removal or arc reversal), which are restricted by both semantic and structural constraints, and selects a new proposed DAG. The scores of both DAGs (old and new) are compared; if the score of the proposed DAG is greater than that of the old DAG, the proposed DAG is accepted as the new DAG, otherwise the proposed DAG is accepted with finite probability:

$$\frac{\pi_{\text{new}}}{\pi_{\text{old}}} = \exp\{[\text{Score}(\mathscr{G}_{\text{new}}) - \text{Score}(\mathscr{G}_{\text{old}})]/T\} \cdot \frac{\text{NumberOfNeighbors}(\mathscr{G}_{\text{old}})}{\text{NumberOfNeighbors}(\mathscr{G}_{\text{new}})}, \tag{7}$$

where $\mathscr{G}_{\text{old}}$ is the old DAG and $\mathscr{G}_{\text{new}}$ is the proposed DAG.

Note that the jumping distribution $\frac{\pi_{\text{new}}}{\pi_{\text{old}}}$ is made symmetric by applying a factor that takes into account the number of neighbors of each DAG.

A key feature of a score function is its decomposability, which means that at each step of the search procedure only the variation in the score given by the perturbation applied on the network needs to be computed. The alternative approach would be to recalculate the complete score at each step, a task that is usually too expensive. Therefore, the ratio described in Eq. (7) is reduced to the difference of the partial scores resulting from the local perturbations at each step of the search procedure. In order to analyze this in greater detail, consider the case of the Info-geo score. The expression for the score at a given step of the search is

$$\begin{aligned} S &= -\log p(D|\hat{\boldsymbol{\theta}}) + \frac{|\boldsymbol{\theta}|}{2}\log\frac{N}{2\pi} + \log \text{vol}(\text{DAG}) \\ &= -\sum_{i=1}^{N}\log p(x_i|x_{\text{pa}(i)},\hat{\boldsymbol{\theta}}) + \sum_{i=1}^{N}\frac{|\boldsymbol{\theta}_i|}{2}\log\frac{N}{2\pi} \\ &\quad + \log \text{vol}(\text{DAG}), \end{aligned} \tag{8}$$

where $\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimator, N is the number of variables in the BN, and log-vol(DAG) is the logarithm of the manifold's volume of the BN with DAG $\mathscr{G}$ (i.e. $\log\int\sqrt{\det \mathbf{I}(\boldsymbol{\theta})}\,\mathrm{d}\boldsymbol{\theta}$, see Section 4).

This shows the decomposability of the score. To simplify the notation, $\mathscr{B} = -\log p(x_i|x_{\text{pa}(i)},\hat{\boldsymbol{\theta}}) + \frac{|\boldsymbol{\theta}_i|}{2}\log\frac{N}{2\pi}$, and $\mathscr{L} = \log \text{vol}(\text{DAG})$. Considering two possible BNs a and b, where b is generated by reversing an arc in a, the scores of a and b can be expressed as

$$S_a = \mathscr{B}_a^{(i)} + \mathscr{B}_a^{(j)} + \sum_{k\neq\{i,j\}}\mathscr{B}_a^{(k)} + \mathscr{L}_a, \tag{9}$$

$$S_b = \mathscr{B}_b^{(i)} + \mathscr{B}_b^{(j)} + \sum_{k\neq\{i,j\}}\mathscr{B}_b^{(k)} + \mathscr{L}_b, \tag{10}$$

where $\mathscr{B}_a^{(i)}$ and $\mathscr{B}_b^{(i)}$ are the partial scores of node i in DAGs a and b; $\mathscr{B}_a^{(j)}$ and $\mathscr{B}_b^{(j)}$ are the partial scores of node j in DAGs a and b; and $\sum_{k\neq\{i,j\}}\mathscr{B}_a^{(k)}$ and $\sum_{k\neq\{i,j\}}\mathscr{B}_b^{(k)}$ are the sums of partial scores of all nodes except i and j in DAGs a and b. Note that after the arc reversal that generates DAG b, X_i and X_j are interchanged. The difference of scores is given by subtracting Eq. (9) from Eq. (10):

$$S_b - S_a = \left(\mathscr{B}_b^{(i)} - \mathscr{B}_a^{(i)}\right) + \left(\mathscr{B}_b^{(j)} - \mathscr{B}_a^{(j)}\right) + (\mathscr{L}_b - \mathscr{L}_a). \tag{11}$$

It can be seen from expression (11) that $\sum_{k\neq\{i,j\}}\mathscr{B}_a^{(k)}$ and $\sum_{k\neq\{i,j\}}\mathscr{B}_b^{(k)}$ cancel out, meaning that the difference between total scores gets reduced to the difference of the partial scores of nodes i and j, plus the difference of log-volumes between both DAGs. (For arc addition and deletion, the difference of scores is $(\mathscr{B}_b^{(i)} - \mathscr{B}_a^{(i)}) + (\mathscr{L}_b - \mathscr{L}_a)$.)

Structural constraints are those associated with the fact that the networks must preserve the condition of being directed acyclic graphs. This means that operations that yield cycles in the network should be excluded as a valid choice. Therefore, at each of the search procedure, an acyclicity check must be performed in order to decide upon the validity of a given perturbation of the network. An acyclicity verification algorithm, see Guidici and Castelo (2003), is used for this purpose. The algorithm is fast, with a time complexity of O(1) for arc additions and of O(n) for arc reversals.

The algorithm applies the semantic constraints implemented by means of the restriction matrix to the list of potential neighbors generated after perturbing the network. The restricted list of neighbors is subsequently used to implement the jumping distribution of the Metropolis–Hastings algorithm.

References

Amari, S.I., 1985. Differential Geometrical Methods in Statistics. Springer-Verlag.

Anderson, R., Vastag, G., 2004. Causal modeling alternatives in operations research: Overview and application. European Journal of Operational Research 156 (1), 92–109.

Baronas, A., Louis, R., 1988. Restoring sense of control during implementation: How user involvement leads to system acceptance. MIS Quarterly 12 (1), 11–23.

Bellman, R., 1961. Adaptive Control Processes: A Guided Tour. Princeton University Press.

Benjamin, R., Levinson, E., 1993. A framework for managing IT-enabled change. Sloan Management Review 34 (4), 23–34.

Berg, L., 1992. The scoop of client–server costs. Computer World 26 (46), 169–176.

Bibi, S., Stamelos, I., 2004. Software process modeling with Bayesian belief networks. In: Proceedings of 10th International Software Metrics Symposium (Metrics 2004) 14–16 September 2004, Chicago, USA.

Brancheau, J., Schuster, L., Mar, S., 1989. Building and implementing an information systems architecture. Data Base 20 (2), 9–17.

Britt, D.A., 1997. Conceptual Introduction to Modeling: Qualitative and Quantitative Perspectives. Lawrence Erlbraum Associates Inc.

Cheney, P., Mann, R., Amoroso, D., 1986. Organizational factors affecting the success of end-user computing. Journal of Mgmt information Systems 3 (1), 65–80.

Chengalur-Smith, I., Duchessi, P., 2000. Client–server implementation: Some management pointers. IEEE Transactions on Engineering Management 47 (1).

Chin, W., 1998. Structural equation modeling in IS research. ISWorld Net Virtual Meeting Center at Temple University, 2–5 November 1998. Available from: <http//interact.cis.temple.edu/~vmc>.

Chwelos, P., Benbasat, I., Dexter, A., 2001. Research report: Empirical test of an EDI adoption model. Information Systems Research 12 (3), 304–321.

Cooper, G., Herskovits, E., 1992. A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9 (4), 309–347.

Cowell, R., Dawid, A., Lauritzen, S., Spiegelhalter, D., 1999. Probabilistic Networks and Expert Systems. Springer-Verlag.

Cox, J., Zmud, R., Clark, S., 1981. Auditing an MRP System. Academy of Management Journal 24 (2), 386–402.

Dolan, R., 1995. The end of delegation? Information technology and the CEO. Harvard Business Review 73 (5), 166–168.

Duchessi, P., Schaninger, C., Hobbs, D., 1989. Implementing a manufacturing, planning and control information system. California Management Review 31 (3), 75–90.

Ettlie, J., Bridges, W., O'Keefe, R., 1984. Organizational strategy and structural differences for radical versus incremental innovations. Management Science 30 (6), 682–695.

Fan, Chin-Feng, Yu, Yuan-Chang, 2004. BBN-based software project risk management. Journal of Systems Software 73, 193–203.

Fenton, N.E., Krause, P., Neil, M., 2002. Software measurement: Uncertainty and causal modelling. IEEE Software 10 (4), 116–122.

Fenton, N.E., Marsh, W., Neil, M., Cates, P., Forey, S., Tailor, M., 2004. Making resource decisions for software projects. In: 26th International Conference on Software Engineering ICSE2004 May 2004, Edinburgh, United Kingdom. IEEE Computer Society, ISBN 0-7695-2163-0, pp. 397–406.

Griffith, T., Northcraft, G., 1996. Cognitive elements in the implementation of new technology: Can less information provide more benefits? MIS Quarterly 20 (1), 99–111.

Guidici, P., Castelo, R., 2003. Improving MCMC model search for data mining. Machine Learning 50, 127–158.

Hamilton, S., Chervany, N., 1981. Evaluating information system effectiveness—Part I: Comparing evaluation approaches. MIS Quarterly 5 (3), 55–69.

Hastings, W., 1970. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57, 97–109.

Heckerman, D., Geiger, D., Chickering, D., 1994. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft Research.

Holland, J., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor.

IBM Consulting Group, 1994. Report on client/server, White Plains, NY.

Jordan, M. (Ed.), 1998. Learning in Graphical Models. MIT Press.

Keil, M., Mann, J., Rai, A., 2000. Why software projects escalate: An empirical analysis and test of four theoretical models. MIS Quarterly 24 (4).

Kiernan, V., 1995. The Impact of Technology on Organizational Transformations. Available from: <http://www.mind-spring.com/~kiernan/mgt6107.html>.

Kirkpatrick, S., Gelatt, D., Vecchi, M., 1983. Optimization by simulated annealing. Science 220, 671–680.

Korb, A., Nicholson, A., 2004. Bayesian Artificial Intelligence. Chapman & Hall/CRC Press, UK.

Lam, W., Bachus, F., 1994. Learning Bayesian networks. An approach based on the MDL principle. Computational Iintelligence 10 (3), 269–293.

Lauría, E., 2005. Learning the structure of a Bayesian network: An application of information geometry and the minimum description length principle. In: Knuth, K.H., Abbas, A.E., Morris, R.D., Patrick Castle, J. (Eds.), Bayesian Inference and Maximum Entropy Methods in Science and Engineering, pp. 293–301.

Lauritzen, S.L., 2000. Causal inference in graphical models. In: Barndorff-Nielsen, O.E., Cox, D.R., Kluppelberg, C. (Eds.), Complex Stochastic Systems. Chapman & Hall, London.

Lauritzen, S., Spiegelhalter, D., 1988. Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society Series B 50 (2), 157–224.

McKeen, J., 1983. Successful development strategies for business application systems. MIS Quarterly 7 (3), 47–65.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., 1953. Equations of State Calculations by Fast Computing Machines. Journal of Chemical Physics 21, 1087–1091.

Neal, R., 1993. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

Neil, M., Fenton, N.E., Nielsen, L., 2000. Building large-scale Bayesian networks. The Knowledge Engineering Review 15 (3), 257–284.

Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman Publishers, San Mateo, CA.

Pearl, J., Verma, T., 1991. A theory of inferred causation. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference. Morgan Kaufmann, San Mateo, CA.

Ramoni, M., Sebastiani, P., 1999. Bayesian methods for intelligent data analysis. In: Berthold, M., Hand, D.J. (Eds.), Intelligent Data Analysis: An Introduction. Springer, New York, NY.

Ravichandran, T., Rai, A., 2000. Quality management in systems development: An organizational system perspective. MIS Quarterly 24 (3).

Reich, B., Benbasat, I., 2000. Factors that influence the social dimension of alignment between business and information technology objectives. MIS Quarterly 24 (1), 81–111.

Rissanen, J., 1986. Stochastic complexity and modeling. Annals of Statistics 14 (3), 1080–1100.

Rissanen, J., 1996. Fisher information and stochastic complexity. IEEE Transactions on Information Theory 42, 40–47.

Robey, D., Farrow, D., 1982. User involvement in information system development: A conflict model and empirical test. Management Science 28 (1), 73–85.

Rodríguez, C., 2001. Entropic priors for discrete probabilistic networks and for mixtures of Gaussian models. In: Mohammad-Djafari, A. (Ed.), Maximum Entropy and Bayesian Methods in Science and Engineering. Available from: <http://arxiv.org/abs/physics/0201016>.

Roweis, S., Saul, L., 2000. Nonlinear dimensionality reduction by locally linear embedding. Science 290, 2323–2326.

Russell, S., Norvig, P., 2003. Artificial Intelligence, A Modern Approach, second ed. Prentice-Hall.

Scheines, R., 1995. Bayesian estimation of Gaussian Bayes networks. In: Proceedings of NIPS 95, Workshop on Learning in Bayesian Networks and Other Graphical Models.

Schultheis, R., Bock, D., 1994. Benefits and barriers to client/server computing. Journal of System Management 45 (2), 12–41.

Schwarz, G., 1978. Estimating the dimension of a model. Annals of Statistics 6, 461–464.

Spirtes, P., Glymour, C., Scheines, R., 2000. Causation, Prediction, and Search, second ed. MIT Press.

Stamelos, I., Angelisa, L., Dimoua, P., Sakellaris, P., 2003. On the use of Bayesian belief networks for the prediction of software productivity. Information and Software Technology 45 (1), 51–60.

Tenenbaum, J., de Silva, V., Langford, J., 2000. A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323.

Witten, I., Frank, E., 2000. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers.

Wixom, B., Watson, H., 2001. An Empirical Investigation of the Factors Affecting Data Warehousing Success. MIS Quarterly 25 (1), 17–41.

Wright, S., Wright, A., 2001. Information system assurance for enterprise resource planning systems: Unique risk considerations. Journal of Information Systems 16, 99–113.