International Journal of Hybrid Information Technology
Vol.2, No.2, April, 2009

Loopy Belief Propagation: Bayesian Networks for Multi-Criteria Decision Making (MCDM)

Wiboonsak Watthayu
Department of Mathematics
King Mongkut's University of Technology Thonburi
Bangkok, Thailand
iwibhayu@kmutt.ac.th

Abstract

Loopy Belief propagation is an increasingly popular method of performing approximate inference on arbitrary graphical models. Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. When used in conjunction with statistical techniques, the graphical model has several advantages for data mining. Influence diagrams provide a compact technique to represent problems of decision making especially multi-criteria decision making (MCDM) under uncertainty. As a number of nodes in the network increases, computing exact solutions and making optimal decision becomes computationally intractable. Approximate solution becomes more efficient in term of the performance of execution and the storage space. In particular, the belief propagation (or sum-product) algorithm has become a well-known means of solving inference problems approximately. Therefore, the loopy belief propagation is the alternative way for approximate solution and is presented in this paper. A solution is approximated where high-probability actions under the policy have a high utility. Actions are then selected which have a high probability under the approximating policy. The loopy belief propagation method is shown to compare favorably to exact methods.

1. Introduction

The Multi Criteria Decision-Making Method (MCDM) involves "making preference decisions (such as evaluation, prioritization, selection, and so on) over the available alternatives that are characterized by multiple, usually conflicting, criteria" [1]. During the past three decades, considerable progress has been made in multiple criteria decision analysis—the modeling and solving of MCDM problems. As electronic commerce becomes increasingly global in recent years, the process of decision-making has become more complicated because there are more alternatives to choose from and more factors to be considered in the process. This requires further advance of decision-making methods and applications for MCDM. The advancement of information technology (IT), which makes globalize e-commerce possible, also provides various technologies for new, and potentially more powerful MCDM methods.

Bayesian network (BN) and influence diagram (ID) are developed in data mining and artificial intelligence (AI) community as principled formalisms for representing and reasoning under uncertainty in intelligent systems. In a BN, entities of interest (e.g., decision criteria and sub-criteria, factors that influence them) are treated as random variables and represented as node in the network, collected by directed arcs indicating probabilistic dependencies between them. The network structure, together with conditional probability tables associated

with each node; provide a compact representation of the joint probability distribution of all variables. A suite of algorithms have been developed for probabilistic inference with BN, especially those which, when some variables' values have been observed, compute the posterior probabilities of others. ID extends BN by adding utility nodes and decision nodes, allowing one to perform various decision related tasks, including computing the expected utility, given observations and decision choices and finding the optimal. BN and ID thus provide a theoretically well-founded and operational basis for modeling MCDM and problem-solving.

This paper focuses on data mining technique that is representing and solving influence diagram (ID) for MCDM problems and presents our ongoing work on developing approximate inference algorithm based on loopy belief propagation algorithm to extend in influence diagrams (ID) for solving MCDM problems.

There are several algorithms, both exact and approximate, for evaluating an influence diagrams, also called inference. Bayesian network inference using the variable elimination algorithm, an exact method, is NP-hard in the worst case but linear in most cases. Even if it is linear, average-sized Bayesian networks take a long time to query. In practice, people often use approximate methods such as Monte Carlo (MCMC) simulation, sampling method, variational method, and so on. Sallans [19] found that the sampling method and MCMC method performed better for larger sample sizes on some networks, however, MCMC and sampling method have a problematic in convergence. They can apply for only in some networks [22].

Loopy belief propagation method is an alternative to evaluate the influence diagram. It has been used for approximate inference in a wide variety of BN model [22]. Loopy belief propagation is fast, deterministic, and can take advantage of structure in the graph to greatly speed up and improve inference. In this paper, the method by using loopy propagation algorithm is proposed to evaluate the approximate inference and influence diagrams. The experimental results of loopy propagation method are compared to others methods such as likelihood weight method, junction tree method and Markov Chain Monte Carlo (MCMC) method. The proposed algorithm shows that it is faster than others (including exact algorithms).

The presentation of this work is organized as follows. In Section 2, the background of MCDM is provided. Bayesian networks and influence diagrams are discussed and a brief review of existing approximate algorithm is also provided in Section 3. Section 4 outlines the way in which loopy belief propagation algorithm can be extended into an ID. Section 5 shows the experimental results. Finally, conclusions and further research are discussed in Section 6.

2. Background

The limitation becomes more apparent in the development of an influence diagrams when we are not able to handle a utility function of a very large network. To overcome this limitation, and maintain the appealing features of probabilistic soundness and graphical nature of BNs, the approximate solution of evaluating ID is developed. There have some approximate algorithms of evaluating IDs such as sampling, and variational methods. However, these algorithms are still face the problem of complexity time when the model becomes large and complicated.

MCDM refers to making decisions in the presence of multiple criteria. In the essence, a MCDM problem is formed into hierarchy composed of four elements: the *goal*, the *objectives*, the *criteria*, and the *alternatives*. These elements can be presented in a matrix format. Let $A=\{A_i,\ldots,A_m\}$ be a set of decision alternatives and $C=\{c_j,\ldots,c_n\}$ a set of criteria according to which desirability of an action is judged. A decision matrix D is an $m\times n$ matrix, in which element d_{ij} indicates the performance of alternative a_i when it is evaluated in terms of decision criteria c_j. It is often assumed that the decision maker has determined the weights of relative importance of the decision criteria, $W=\{w_j,\ldots,w_n\}$ [10]. The total score for each alternative is obtained by the following formula:

$$A_i=\Sigma_j w_j d_{ij}$$

When the overall scores are calculated for all the alternatives, the one with the highest score is chosen. During the past three decades, along with the rapid development of information technologies, considerable progress has been made in multiple criteria decision analysis—the modeling and solving of MCDM problems, and many MCDM methods have been developed (see [3] for a survey). However, some of the existing methods have been criticized as ad hoc and, to certain degree, unjustified on theoretical and/or empirical grounds.

3. Bayesian Network and Influence Diagram

3.1 Bayesian Network

Bayesian networks (BN), are widely used for knowledge representation and reasoning under uncertainty in intelligent systems [2 & 4]. In a general form, the structure of a BN is a directed acyclic graph (*DAG*) in which nodes correspond to random variables of interest and directed arcs represent *direct* causal or influential relation between nodes. The uncertainty of the interdependence of the variables is represented locally by the conditional probability table (*CPT*) $\Pr(x_i\mid\pi_i)$ associated with each node x_i, where π_i is the parent set of x_i. An independence assumption is also made with BN that x_i, given its parents π_i, is independent of any other variables except its descendents. The graphical structure of BN allows an unambiguous representation of interdependency between variables. This, together with the independence assumption, leads to one of the most important features of BN that the joint probability distribution of $X=(x_i,\ldots,x_n)$ can be factored out as a product of the conditional distributions in the network,

$$\Pr(X=x)=\prod_{i=1}^{n}\Pr(x_i\mid\pi_i).$$

Figure 1 below gives an example BN, “Family Out”, where *Family-out* and *Bowel-problem* are direct causes for dog going out, and *Dog-out* directly affects whether we hear barking or not. Once we know whether the dog is out or not, the probability of our hearing barking has nothing to do with Family-out and Bowel-problem because their influence on hearing barking is blocked by the instantiation of *Dog-out*. Each node in the network has a conditional probability table (CPT). Each column in the table contains the conditional probability of each node value for a possible combination of values of the parent nodes.

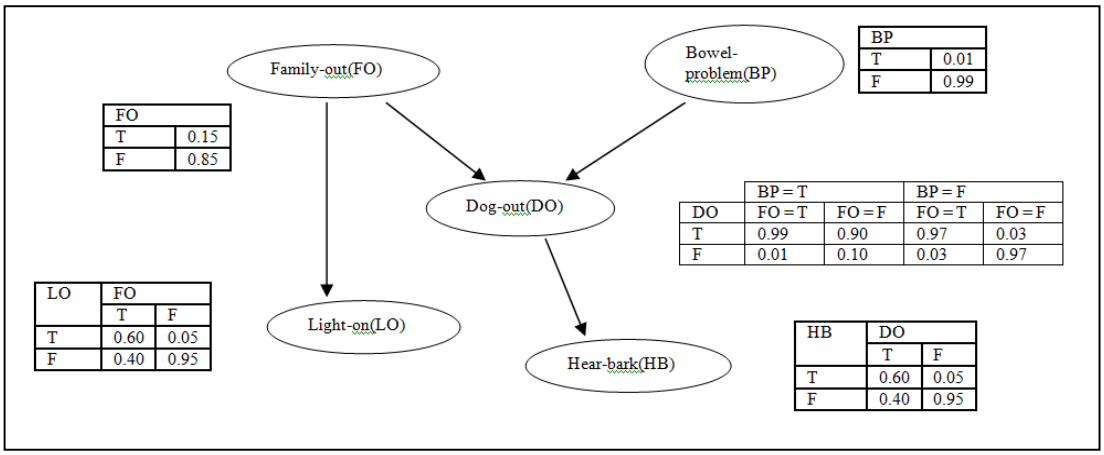

Figure 1. The "Family out" Bayesian Network (Charniak, 1991). Various probabilistic queries can be answered. For example, P(HB), the prior probability of hearing barks is computed as 0.148, and the posterior probability P(FO=T|HB=T ∧ LO=F) = 0.689.

With the joint probability distribution, BNs can support in theory any probabilistic inference in the joint space. Moreover, although it has been proven that inference in BN with general *DAG* structure is *NP*-hard [3], probabilistic inference algorithms that are more efficient than the brute force use of the gigantic joint probability table have been developed by exploring the interdependency captured by the network structure. Most important among them are algorithms for computing posterior probabilities $\Pr(x_i \mid e)$, where *e* denotes evidence, the observed values for some variables. These class of algorithms include "belief propagation" [2] and "Junction tree" [12, 13, & 14] for exact solutions, and various statistical sampling techniques (e.g., Markov Chain Monte Carlo sampling) for approximate solutions with extremely large BN (see [15] for a detailed explanation of the most commonly used BN inference algorithms).

3.2 Influence Diagram

Influence diagrams for solving decision problems extend BN with two additional types of nodes, namely *decision nodes* and *utility nodes*. Nodes for the random variables in the BN are called *chance nodes* in ID. A decision node defines the action alternatives considered by the user. Every decision node has a finite number of alternatives standing for the actions that the decision maker will take to achieve the desired outcome. A decision node is connected to those chance nodes whose probability distributions are directed affected by the decision. A utility node is a random variable whose value is the utility of the outcome. Like other random variables, a utility node holds a table of utility values for all configurations of its parent nodes. In an influence diagram, the value of each decision variable is not determined probabilistically by its predecessors, but rather is imposed from the outside to meet some optimization objective [6].

In an ID, let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of mutually excusive actions, and H be the determining variable. A utility table *U(A,H)* is need to yield the utility for each configuration of action and determining variable in order to decide between the actions

in *A*. The problem is solved by calculating the action that maximizes the expected utility:

$$EU(a) = \prod_H U(a,H)P(H \mid a),$$

where $U(a,H)$ are the entries of the utility table in the value node *U*. The conditional probability $P(H|a)$ is can be computed from CPT of the variable $h_i \in H$, given the action *a* is fired.

Figure 2 below represents an ID about weather and a decision to carry an umbrella [6]. *Forecast* and *Weather* are chance nodes containing the probabilistic information about the weather and forecast. *Satisfaction* is a utility or value node. *Umbrella* is a decision node. The objective is to maximize expected Satisfaction by appropriately selecting values of Umbrella for each possible forecast. In addition to probabilities, the values of Satisfaction for each combination of Umbrella and Weather are also given.

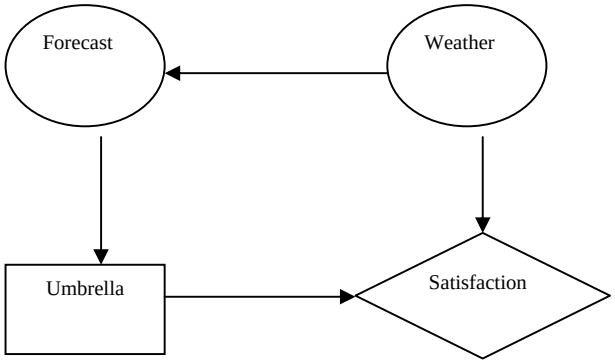

Figure 2. An example of influence diagram [6]

The algorithm for evaluating an ID goes as follows [4]: (1) set the evidence variables for the current state, (2) for each possible value of the decision node, set the decision node to that value; (3) calculate the posterior probabilities for the parent nodes of the utility node using a standard probabilistic inference algorithm, and (4) calculate the resulting utility function for the action and return the action with the highest utility.

3.3 Inference Algorithms

The algorithms that propagate beliefs and update the beliefs in each node can be exact algorithms (for simple structures) or approximate algorithms (for complex structures); a detailed explanation of the most common algorithms can be found in Castillo [15]. We briefly review some of the commonly used exact and approximate algorithm.

3.3.1 Exact Inference Algorithms

The most commonly used exact algorithm is junction tree based algorithm [12, 13, & 14]. The basic data structure used by these algorithms is called a junction tree, which can be constructed via the process of moralization and triangulation [13 & 14]. Every node in junction tree, which is a clique of the moralized and triangulated BN, has its

internal states. It can send or receive message from its neighbors. Inference in BN is then formulated in terms of message passing in the junction tree. The message exchange is based on the constraints: (1) each node sends one message to each neighbor, and (2) each node can send a message to a neighbor after it has received a message from its all other neighbors. A message sent by a node is prepared on the basis of the messages received and its internal state.

3.3.2 Approximate algorithms

Approximate BN inference algorithms include stochastic simulation algorithms, model simplification methods, search-based methods and loopy belief propagation.

Stochastic simulation algorithms, also called stochastic sampling or Monte Carlo algorithm, are the most well known approximate BN inference algorithms. They generate a set of randomly selected samples or instantiations of the network according to the CPTs in the model, and then approximate probabilities of query variables by the frequencies of appearances in the sample. Stochastic simulation algorithms are big class. They can be divided into two main categories: importance sampling algorithms and Markov Chain Monte Carlo (MCMC) methods.

The most efficient importance sampling algorithm reported so far seems to be Jian Cheng's adaptive importance sampling for BNs (AIS-BN) (Cheng & Druzdzel, 2000). AIS-BN reduces the sampling variance by learning a sampling distribution that is as close as possible to the optimal importance sampling function. AIS-BN algorithm introduces different weights for samples generated at different learning stages. In Cheng (Cheng & Druzdzel, 2000), AIS-BN is combined with new stopping rules to yield two other sampling algorithms that work well for large networks.

Another group of stochastic sampling algorithms is MCMC method. The basic idea is that we can approximate the distribution by generating independent sampling from it. Then we can estimate the value of a quantity relative to the original distribution by computing its value relative to our samplings. MCMC is based on the theory of Markov chains. A Markov chain is a stochastic process, evolving in discrete time steps. X_0, X_1, ..., X_N are denoted the state of the process at the various time steps. The distribution for X_0 is the initial distribution. The distribution of X_{n+1} given the past depends only on X_n. The conditional distribution $P(X_{n+1} \mid X_n)$ is called the transition probability. If the transition probability does not depend on n then the process is called a stationary process. MCMC method uses Markov Chains in order to generate the samples for approximate inference.

A final class of algorithm for performing approximate inference is provided by loopy propagation methods. Loopy propagation applies Pearl's polytree propagation algorithm in BNs with loops. Researchers have empirically demonstrated that loopy belief can perform well in the context of error-correcting codes and computer vision [24]. But for some other graphs with loops, loopy propagation may give poor results or even fail to converge. More recently, it has been shown that there is a close connection between the belief propagation algorithm and certain approximations to the variational free energy in statistical physics [25].

4. Extended ID with Loopy Belief Propagation

In loopy belief propagation, Pearl's algorithm works similarly for polytrees or network containing no loops. Given a node *X* having parents $U_1,...,U_m$ and children $Y_1,...,Y_n$, the evidence will be represented with *E*, with evidence "above" *X* written as e^+ and evidence "below" *X* written as e^-. Evidence can flow both down the network (from parent to child) or up the network (child to parent). The messages are labeled as π and λ messages, respectively. We need to compute the belief of *X* having the value of true or false (*BEL*(*X* = true)/*BEL*(*X* = false)). For example, if *X* is true, *BEL*(*X*) = 1.0. The belief of *X* given its parent and child values is

$$BEL(x) = P(X \mid e) = \alpha\lambda(X)\pi(X) \quad (1)$$

Where α is a normalizing value, $\lambda(X) = P(e^- \mid x)$, and $\pi(X) = P(x \mid e^+)$.

$$\lambda(X) = \prod_{i=1}^{j} \lambda_{Y_i} \quad (2)$$

$$\pi(X) = \sum P(x \mid \vec{u})\pi(\vec{u}) \quad (3)$$

U is a vector of possible values for $U_1,...,U_m$. Once *BEL*(*X*) has been updated, *X* must send π and λ messages to its' children and parents, respectively. But we must ensure to update Y_i's belief based on evidence from Y_i and likewise for U_i. This is done by calculating the messages to send to Y_i and U_i as follows:

$$\pi(Y_i) = \frac{BEL(X)}{\lambda_{Y_i}} \quad (4)$$

and

$$\lambda(U_i) = \beta \sum_X \lambda(X) \sum_{u_k : k \neq i} P(X \mid \vec{u}) \prod_{k \neq i} \pi(u_k) \quad (5)$$

Where β is a normalizing factor and $\vec{u}$ is the vector of values for $U_1,...,U_m$. Therefore, each node tells its neighbors what it believes, based on prior probability and the belief of its neighbors. For more details of Pearl's algorithm, see [8].

The basic step for this algorithm as follows:

1. Initialize the network

2. Update beliefs

3. Propagate changes in belief

4. If beliefs changed, loop on step 2

In the analysis, for loopy belief propagation, we look at a network with a single loop first. Given evidence somewhere on the loop, the evidence will be double-counted. In other words, the beliefs of node will propagate around the loop in both directions. This will obviously give us incorrect belief for any given node on the loop. But it has been shown that for the graphs with a single loop [12].

1. Unless all the compatibilities and deterministic, loopy belief propagation will converge.

2. An analytic expression relates the correct marginal to the loopy marginal

3. If the hidden nodes are binary, then the loopy beliefs and the true beliefs are both maximized by the same assignments, although the confidence that assignment is wrong for the loopy beliefs.

Formally, ID is defined by $ID = (X, D, P, R)$, where $X = \{X_1, ..., X_n\}$ is a set of chance variables and $D = \{D_1, ..., D_m\}$ is a set of decision nodes [23]. The chance variables are further divided into observable or unobservable. The discrete domains of decision variables denote its possible set of action. An action in the decision node D_i is denoted by d_i. Every chance node X_i is associated with a CPT, $P_i = \{P(X_i \mid pa_i)\}$, $pa_i < X \cup D - \{X_i\}$. Each decision variable D_i has a parent set $pa_{D_i} \subseteq X \cup D$ denoting the variables, whose values will be known and may directly affect the decision. The reward functions $R = \{r_1, ..., r_j\}$ are defined over subsets of variables $Q = \{Q_1, ..., Q_j\}$, $Q_i \subseteq X \cup D$, called scopes, and the utility function is defined by $u(x) = \sum_j r_j(x_{Q_j})$.

Let $D_1, ..., D_m$ be the decision variables in ID. A decision rule for a decision node D_i is a mapping

$\delta_i : \Omega_{pa_{D_i}} \rightarrow \Omega_{D_i}$, where for $S \subseteq X \cup D$, Ω_i is the cross product of the individual domains of variables in *S*. A policy is a list of decision rules $\Delta = \{\delta_1, ..., \delta_n\}$ consisting of one rule for each decision variable [23]. To evaluate an influence diagram is to find an optimal policy that maximize the expected utility (MEU) and to compute the optimal expected utility. Assume that x is an assignment over both chance variables and decision variables $X = (x_1, ..., x_n, d_1, ..., d_m)$, the MEU task is to compute

$$E = \max_{\Delta=(\delta_1,...,\delta_m)} \sum_{x_1,...,x_n} \prod_{x_i} P(x_i, e \mid x^{\Delta}_{pa_i}) u(x^{\Delta}) \qquad (6)$$

where x^{Δ} denotes an assignment $X = (x_1, ..., x_n, d_1, ..., d_m)$ where each d_i is determined by $\delta_i \in \Delta$ as a function of $(x_1, ..., x_n)$. Namely: $d_i = \delta_i(x)$.

In ID, when loopy belief propagation is applied, the procedure of propagation and update beliefs are the main part of the process of evaluating ID.

For the update and propagate belief, we focus on

1. If x_p ($p = 1,2,\ldots,n$) is a chance variable, it will computed
   $\lambda_p = \sum_{x_p} \prod_i \lambda_i$ and
   $\Theta_p = \frac{1}{\lambda_p} \sum x_p \prod_{i=1}^{j} \lambda_i \sum_{j=1}^{l} \Theta_j$

   where $\Theta_1, ..., \Theta_n$ is a utility components.
2. If x_p is a decision variable, it will computed
   $\lambda_p = max_{x_p} \prod_{i=1}^{j} \lambda_i \sum_{j=1}^{i} \Theta_j$ or
   $\Theta_p = max_{x_p} \sum_j \Theta_j$

Finally, λ_p will be added with the largest index variable in the scope of Θ_p and decision will compute from the largest Θ_p for the maximum expected utility.

5. Experiments and Results

The Bayes Net Toolbox was selected for the experiments because it includes several inference algorithms. The BNT written in MATLAB is also used in experiments [22].

The number of nodes in Bayesian network were randomly created from 5, 10, …, 50. The experiments and results were presents in the graph: figures 3 shows the relative error in order to check how closed the exact solution can be obtained and figure 4 shows the average running time for queries of IDs.

The experiments was run trials for both exact and approximate algorithms using some IDs, one tiny (10 nodes) and the other medium-sized (20 nodes) and other large-sized (50 nodes). First we tested with three algorithms: several exact inference algorithms (e.g. likelihood weighting and junction tree), and loopy propagation for the running time results.

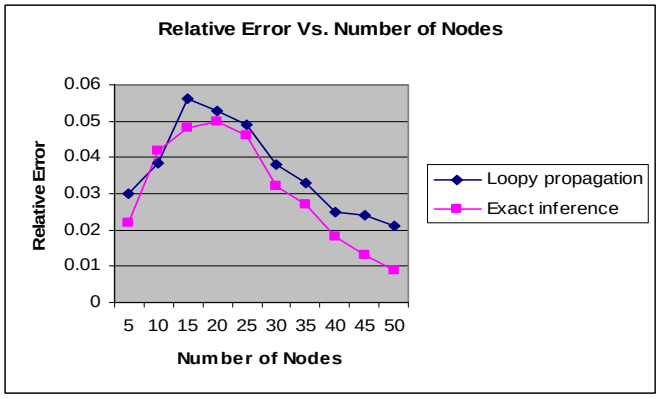

Figure 3. The Relative errors vs. number of nodes

As can be seen from the Figure 4, the exact inference had good speedup with small number of node and can produce the exact solutions with large number of nodes, but not with large number of nodes when number of nodes larger than 30 (30 nodes up). It found that an approximate algorithm is good in running time with a large network (20 nodes up).

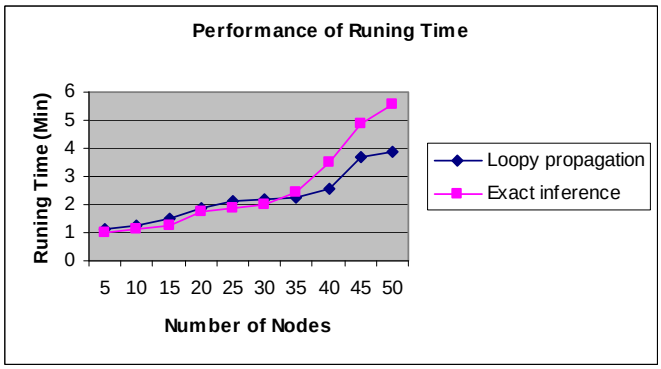

Figure 4. The running time with number of nodes

Then the set of tests were conducted on 10 randomly generated influence diagrams. We supposed that each of influence diagrams had a single utility node with all of the decision nodes and chance nodes. The mean results and 95% confidence intervals on the mean are shown in Figure 3 for different numbers of nodes. Relative error was computed as

$RE = \frac{(U_{opt} - U_{appx})}{U_{opt}}$, here U_{opt} and U_{appx} are the utilities achieved with the optimal and approximated actions respectively. On these influence diagrams a uniform random policy has a relative error of approximately 0.5 [19].

It noticed that in single utility node case, the loopy belief propagation converged after less than five iterations on average. The exact inference methods converged more than five iterations. However, the utility achieved by the loopy belief propagation method depends on the number of samples taken performance matched the other methods.

For the relative error of utility, the approximate algorithm (loopy belief propagation method) and the exact algorithm, the approximate method is uniformly worse, however the error decrease for large numbers of nodes. The solution with approximate algorithm suffers as compared to exact algorithms, with performance dropping in the worst cast for N = 15. In the best case, the decrease in performance is for N = 50. In general, the performance of the loopy belief propagation improves as number of nodes increase.

The loopy belief propagation method can be converged faster than other methods; however, it can be problematic when the some large networks in real world decision problems are applied. Further research will be investigated.

6. Conclusion

In this paper we presented the developing a methodology for approximated solution in evaluating of influence diagrams. The loopy belief propagation algorithm is extended to influence diagram for solving the approximated solutions. In the exact methods, likelihood weighting had good speedup with small number of node and can produce the exact solutions, but not with large number of nodes (30 nodes up). We found that approximate algorithm is good in running time with a large network; however we can get only the approximated solutions.

The experimental results indicated that loopy belief propagation is a competitive technique, especially when the networks are very large. Although the exact methods are guaranteed to get the right answer in the limit of number of nodes, they can be required a lots of memory and for the best quality of computational performance. On the other hand, approximate methods can be a good choice in term of a large network and best running time for the computational. In addition, in real world decision problem, the network might be very large and complex, therefore, if we want to get the approximate solution with the fast running time. The loopy belief propagation can be a good candidate especially in MCDM problems.

Various BN inference algorithms are readily available and can be directly used for typical decision tasks such as recommending the optimal solution. Work presented here is only the first step of our effort toward a comprehensive solution to this very complex problem. Several issues need to be addressed in order to transform our framework from conceptual to one that is workable in real world situations. First, we will investigate using machine learning techniques for influence diagram construction with more than one number of utility nodes. Secondly, we will extend our experiment to multiple utility nodes for each random influence diagram. The third issue concerns the computational

complexity of BN inference, which has been shown to be NP-hard (e.g., it may in worst case take time exponential to the network size, and thus computationally intractable for MCDM problems involving large number of elements). A promising approach is to combine the techniques of autonomous software agents and multi-sectioned BN to decompose a large BN into smaller subnets, each of which is managed by an agent [18]. We also plan to experiment with various approximate inference algorithms to see if they provide acceptable accuracy.

References

[1] Hwang, C. L. and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications*, Springer-Verlag, Berlin, 1981.
[2] Pearl, J., (1988), *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, ISBN 1-55860-479-0, 1988.
[3] Stewart T. J., "A critical survey on the status of multiple criteria decision making theory and practice," OMEGA International, 1992.
[4] Russell, S., and Norvig, P., *Artificial Intelligence: A modern approach*. Prentice Hall, ISBN 0-13-103805-2, 1995.
[5] Cooper, G.F., "The computational complexity of probabilistic inference using Bayesian belief networks", Artificial Intelligence, 42, p393-405, 1990.
[6] Jensen, Finn V., "Cautious Propagation in Bayesian Networks." Proceedings of the Eleventh Confernece on Uncertainty in AI (UAI-95), Morgan Kaufman, San Mateo, CA, 323-328, 1995.
[7] Vincke, P., Multi-*criteria Decision-Aid*, Wiley, Chichester, 1992.
[8] Chan, H. and Darwiche, A., "When do Numbers Really Matter?" , Journal of AI Research, 2002.
[9] Norsys, *Netica Application*, Norsys Software Corp., http://www.norsys.com, 1998.
[10] Zimmermann, H.-J., Fuzzy Set Theory and Its Applications, Kluwer Academic Publishers, Third Revised Edition, Boston, MA, USA, 1996.
[11] Fenton, N. and Neil, M., "Making Decisions: Using Bayesian Nets and MCDA," Computer Science Department, Queen Mary and Westfield College, London, 2000.
[12] Shafer, G., *Probabilistic Expert Systems*, CBMS-NFS regional Conference Series in Applied Mathematics; SIAM, 1996.
[13] Jensen, Finn V., Olesen, K. G., and Anderson, S. K., "An Algebra of Bayesian Belief Universe for Knowledge-based System": In Networks, 20:637-660, 1990.
[14] Madson, A. L. and Jensen, F. V., "Lazy propagation in junction trees", in proceedings of the 14th Conference on uncertainty in Artificial Intelligence, 1998.
[15] Castillo, E., Gutierrez, J. M., and Hadi, A. S., *Expert Systems and Probabilistic Network Models*: Springer, 1997.
[16] Murphy, K. P., "Software Packages for Graphical Models: Bayesian Networks", http://www.cs.berkeley.edu/~murphyk/Bayes/bnsoft.html, 31 March 2007.
[17] Neapolitan, R. E., *Learning Bayesian Networks*, Pearson Prentice Hall, 2004.
[18] Xiang, Y., *Probabilistic Reasoning in Multiagent Systems: A Graphical Model Approach*, Cambridge University Press, 2002.
[19] Sallans, B., "*Variational Action Selection for Influence Diagrams*," OEFAI-TR-2003-29, 2003.
[20] Charnes , J. M. and P. P. Shenoy, "A forward Monte Carlo method for solving influence diagrams using local computation. School of Business Working Paper No. 273, School of Business, University of Kansas, 1999.
[21] Ortiz, L. and L. P. Kaelbling, " Sampling methods for action selection in influence diagrams, in Proc. Seventeenth National Conference on Artificial Intelligence, 2000.
[22] Murphy, K. P., Weiss, Y. and Jordan, M., "Loopy Belief Propagation for Approximate Inference: An Empirical Study", UAI'99, 1999.

[23] Dechter, R., "A New Perspective on Algorithms for Optimizing Policies under Uncertainty", AAAI'00, 2000.
[24] Mackay, D. J., McEliece, J., and Cheng, J. F. "Turbo decoding as an instace of pearl's belief propagation algorithm". IEEE Journal of Selected Areas of Communication, p. 140-152, 1998.
[25] Weiss, Y. and Freeman, W. "Correctness of belief propagation in Gaussian graphical models of arbitrary topology." Neural Computation, 2006.

Authors

Dr. Wiboonsak Watthayu is a lecturer at department of Mathematics, King Mongkut's university of technology of Thonburi (KMUTT), Bangkok, Thailand. His research area is data mining and decision making modeling.