

Efficient Sensor Selection for Active Information Fusion

Yongmian Zhang, *Member, IEEE*, and Qiang Ji, *Senior Member, IEEE*

Abstract—In our previous paper, we formalized an active information fusion framework based on dynamic Bayesian networks to provide active information fusion. This paper focuses on a central issue of active information fusion, i.e., the efficient identification of a subset of sensors that are most decision relevant and cost effective. Determining the most informative and cost-effective sensors requires an evaluation of all the possible subsets of sensors, which is computationally intractable, particularly when information-theoretic criterion such as mutual information is used. To overcome this challenge, we propose a new quantitative measure for sensor synergy based on which a sensor synergy graph is constructed. Using the sensor synergy graph, we first introduce an alternative measure to multisensor mutual information for characterizing the sensor information gain. We then propose an approximated nonmyopic sensor selection method that can efficiently and near-optimally select a subset of sensors for active fusion. The simulation study demonstrates both the performance and the efficiency of the proposed sensor selection method.

Index Terms—Active information fusion, Bayesian networks (BNs), sensor selection, situation awareness.

I. INTRODUCTION

INFORMATION fusion is playing an increasingly important role in improving the performance of sensory systems for various applications, including situation assessment, enemy intent understanding and prediction, and threat assessment. As sensors become ubiquitous, persistent, and pervasive, and coupled with the ever increasing demand for less time and fewer resources, it becomes critically important to perform selective fusion so that decision can be made in a timely and efficient manner. The need for sensor selection is further demonstrated by the availability of an increasingly large volume of sensory data and by the variability of sensor reliability over time and over location. It is important to select the sensors not only to reduce the amount of data to integrate but also to improve fusion accuracy by selecting the most reliable sensors for a certain location at a certain time, by selecting complementary sensors, and by reducing sensor redundancy. Active fusion serves these purposes well. Active fusion extends the paradigm of informa-

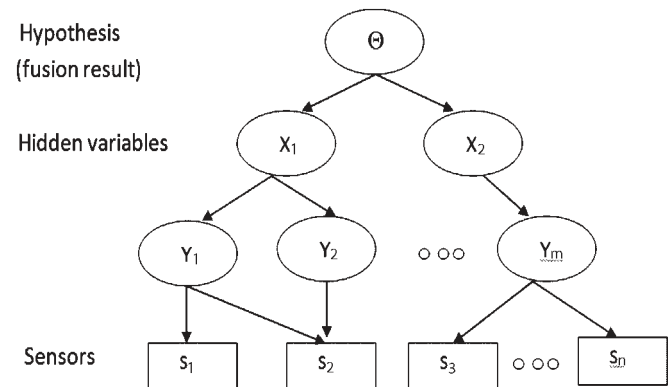


Fig. 1. BN is used for active information fusion, where Θ and S_i are hypothesis and sensors, respectively. X_i and Y_i are the intermediate variables, and they are needed to model the relationships among sensors and the hypothesis. Sensor fusion is accomplished through probabilistic inference given the sensory measurements.

tion fusion by being not only concerned with the methodology of how to combine information but also concerned with the fusion efficiency, timeliness, and accuracy. Active fusion can be defined as the process of combining data with a control mechanism that dynamically selects a subset of sensors to minimize uncertainty in situation assessment and to maximize the overall expected utility in decision making.

In our previous work [1], we formalized an information fusion framework based on Bayesian networks (BNs) to provide active and sufficient information fusion. BNs are used to model a number of uncertain events, their spatial relationships, and the sensor measurements. Given the sensory measurements, information fusion is performed through probabilistic inference using the BN. This can be accomplished through bottom-up belief propagation, as illustrated in Fig. 1. Our previous work, however, did not address the core issue in active fusion, i.e., efficient sensor selection. This is the focus of this paper.

Based on information theory [2], the more sensors¹ we use, the more information we can obtain. However, every act of information gathering incurs cost. Sensor costs may include physical costs, computational costs, maintenance costs, and human costs (e.g., risk). Many applications are often constrained by limited time and resources. An essential issue for active information fusion is to select a subset of the most

Manuscript received May 20, 2008; revised October 25, 2008, December 19, 2008, and March 11, 2009. First published October 20, 2009; current version published June 16, 2010. This work was supported in part by the Air Force Office of Scientific Research under Grant F49620-03-0160. This paper was recommended by Associate Editor R. Lynch.

The authors are with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: zhangy@ecse.rpi.edu; qji@ecse.rpi.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSMCB.2009.2021272

¹For generality, sensors could refer to any devices/means of acquiring information. For example, they may be electromagnetic or acoustic devices or they could also be direct observations of the world through reconnaissance and intelligence gathering activities.

synergetic sensors, which can maximally reduce the uncertainty about the events of interest with minimum costs. Dynamically determining the best set of sensors, given the uncertainty about the state of the world, requires to enumerate all the possible subsets of sensors, which is computationally intractable and practically infeasible. This computational difficulty is twofold. First, the computation of a sensor selection criterion such as mutual information is exponential with respect to the number of sensors. Second, searching for an optimal subset of sensors is also NP-hard, since the sensor space exponentially increases with the number of sensors. To address this computational difficulty, a common practice is to use myopic analysis, which assumes that only one observation will be available at a time, even when there is an opportunity to make a set of observations [3]–[6]. There is a vast literature on the problem of single optimal sensor selection [7]–[9]. However, the myopic approach cannot guarantee to obtain the best evidences that most effectively reduce uncertainty and cost. To effectively reduce uncertainty and cost, one should use nonmyopic selection, which simultaneously considers several observations before making a decision. The most common nonmyopic method is the greedy approach. While efficient, it cannot guarantee optimality with the selected sensors. Other works try to overcome the limitations with the greedy approach, yet with their own strong assumptions. In [10], Heckerman *et al.* presented an approximate nonmyopic approach based on the central-limit theorem in an influence diagram (ID) for efficiently computing the value of information. Their method, however, assumes that the sensors are conditionally independent of each other, given the decision variable, and that the decision variable is binary. Krause and Guestrin [11], [12] presented a randomized approximation algorithm for selecting a near-optimal subset of observations for graphical models. Under the assumptions that the sensors are conditionally independent given the decision variables, the information gain is then guaranteed to be a submodular function, and the theory of submodular functions can then be applied to achieve a near-optimal solution in selecting a subset of observations using a greedy approach. Recently, Liao and Ji [13] have presented an approximation algorithm for the nonmyopic computation of the value of information in an ID. Their method extends the approach in [10] without requiring the sensors being conditionally independent of each other and the decision node being binary.

This paper takes another avenue of approach to efficiently select a subset of near-optimal sensors without the strong sensor independence assumptions, as made in [10] and [12]. Specifically, we first introduce a new quantitative measure of sensor synergy based on mutual information. Based on the synergy measure, we then introduce a method to efficiently compute the least upper bound (LUB) of mutual information for a set of sensors. Experiments show that the LUB closely approximates the mutual information in value, as shown in Figs. 5 and 6. Hence, the computational difficulty with computing the exact nonmyopic mutual information can, therefore, be circumvented by computing its LUB instead. In addition, the synergy measure can also be used to prune the sensor space, which, therefore, reduces the search time for the best sensor set. A summary of the mentioned work may be found in [14].

II. PROBLEM FORMULATION

The problem of sensor selection for active fusion can be stated as follows: Assume that there are m sensors S_i , $i = 1, \dots, m$, available that provide measurements of the world. Let Θ be a set of hypothesis θ_k of the world situation $k = 1, \dots, K$. Let $\mathbf{S} = \{S_1, \dots, S_n\}$ be a subset of n sensors selected at time t , where $n \in \{1, \dots, m\}$. Let $C(\mathbf{S})$ be the cost to use the set of sensors \mathbf{S} . The objective of sensor selection at time $t + 1$ is to select a subset of sensor \mathbf{S}^* to achieve the maximal utility, i.e.,

$$\mathbf{S}^* = \arg \max_{\mathbf{S} \in \mathcal{S}} U(u_1, u_2) \quad (1)$$

where u_1 and u_2 denote information gain (i.e., the mutual information) and the sensor usage cost saving, respectively, \mathcal{S} represents all the possible subsets of sensors, and $U(u_1, u_2)$ is a utility function. Here, we use $u_2 = 1 - C(\mathbf{S})$ to convert the sensor usage cost to the corresponding cost saving, which makes u_1 and u_2 qualitatively equivalent. For simplicity, in this paper, we assume that the cost is the same for all sensors. Hence, we can ignore u_2 .

The major difficulty of using (1) for sensor selection is to efficiently compute the information gain u_1 . From information theory, the entropy of hypothesis Θ given a sensor S_i measures how much uncertainty exists in Θ given S_i , i.e.,

$$H(\Theta | S_i) = - \sum_{s_i} \sum_{\theta} P(\theta, s_i) \log P(\theta | s_i) \quad (2)$$

where s_i denotes a reading of sensor S_i .² Subtracting $H(\Theta | S_i)$ from the original uncertainty in Θ without S_i , i.e., $H(\Theta)$, yields the expected amount of information about Θ that S_i is capable of providing

$$\begin{aligned} I(\Theta; S_i) &= H(\Theta) - H(\Theta | S_i) \\ &= - \sum_{\theta} P(\theta) \log P(\theta) \\ &\quad + \sum_{s_i} \left\{ P(s_i) \sum_{\theta} P(\theta | s_i) \log P(\theta | s_i) \right\} \\ &= \sum_{\theta} \sum_{s_i} P(\theta, s_i) \log \frac{P(\theta | s_i)}{P(\theta)} \end{aligned} \quad (3)$$

where $I(\Theta; S_i)$ is referred to as the mutual information, which characterizes the expected total uncertainty-reducing potential of Θ due to S_i . The mutual information for a sensor set $\mathbf{S} = \{S_1, \dots, S_n\}$ can be obtained by

$$\begin{aligned} I(\Theta; \mathbf{S}) &= H(\Theta) - H(\Theta | \mathbf{S}) \\ &= - \sum_{\theta} P(\theta) \log P(\theta) \\ &\quad + \sum_{\theta} \sum_{s_1} \dots \sum_{s_n} \{ P(\theta, s_1, \dots, s_n) \log P(\theta | s_1, \dots, s_n) \} \\ &= \sum_{\theta} \sum_{s_1} \dots \sum_{s_n} \left\{ P(\theta, s_1, \dots, s_n) \log \frac{P(\theta | s_1, \dots, s_n)}{P(\theta)} \right\} \end{aligned} \quad (4)$$

²Without loss of generality, here we assume discrete sensor measurement. The theories can be straightforwardly extended to continuous sensor measurements.

where $P(\theta, s_1, \dots, s_n)$ and $P(\theta | s_1, \dots, s_n)$ at time t can directly be obtained through BN inference. The mutual information in (4) provides a sensor selection criterion in terms of the uncertainty reduction potential, i.e., mutual information.

It is clear from (4) that when the number of sensors in \mathbf{S} is large or when the number of states for each sensor is large, it becomes computationally impractical to simply implement this information-theoretic criterion, because it generally requires time exponential in the number of summations to exactly compute the mutual information. The remainder of this paper addresses this computational difficulty.

III. APPROXIMATION ALGORITHM

In this section, we give a graph-theoretic definition of sensor synergy. We then present the theorems on which our algorithm is based.

A. Sensor Synergy in Information Gain

Throughout this section, it is assumed that we have obtained $I(\Theta; S_i, S_j)$ and $I(\Theta; S_i)$, i.e., the mutual information of all pairs of sensors and individual sensors with respect to Θ , respectively. We will introduce an efficient method to obtain all $I(\Theta; S_i, S_j)$ in Section III-C. We first define a synergy coefficient to characterize the synergy between two sensors, and then extend this definition to multiple sensors.

Definition 1 (Synergy Coefficient): A measure of the expected synergetic potential between two sensors S_i and S_j in reducing the uncertainty of hypothesis Θ is defined as

$$r_{ij} = \frac{I(\Theta; S_i, S_j) - \max(I(\Theta; S_i), I(\Theta; S_j))}{H(\Theta)}. \quad (5)$$

The denominator $H(\Theta)$ in (5) is to restrict r_{ij} to the interval $[0, 1]$. It can easily be proved that $r_{ij} \geq 0$ based on the ‘‘information never hurts’’ principle [2], i.e., $I(\Theta; S_i, S_j) \geq I(\Theta; S_i)$, and $I(\Theta; S_i, S_j) \geq I(\Theta; S_j)$. This follows that S_i and S_j taken together are always more informative than when they are taken alone. The larger r_{ij} is, the more synergetic the sensors S_i and S_j are. Obviously, $r(\cdot, \cdot)$ is symmetrical in S_i and S_j , and $r_{ij} = 0$ if $i = j$.

Definition 2 (Synergy Matrix): Let a sensor set be $\mathbf{S} = \{S_1, \dots, S_n\}$. The sensor synergy matrix is an $n \times n$ matrix defined as

$$R = \begin{bmatrix} 0 & r_{12} & \cdots & r_{1n} \\ r_{21} & 0 & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & 0 \end{bmatrix}. \quad (6)$$

R is an information measure of synergy among sensors that is based on pairwise sensor synergy. With a synergy matrix, we naturally define its graphical representation.

Definition 3 (Synergy Graph): Given a sensor synergy matrix, a graph $G = (\mathbf{S}, \mathbf{E})$, where \mathbf{S} 's are the nodes representing the set of available sensors, and \mathbf{E} 's are the links representing the set of pairwise synergetic links weighted by synergy coefficients r_{ij} , is a sensor synergy graph.

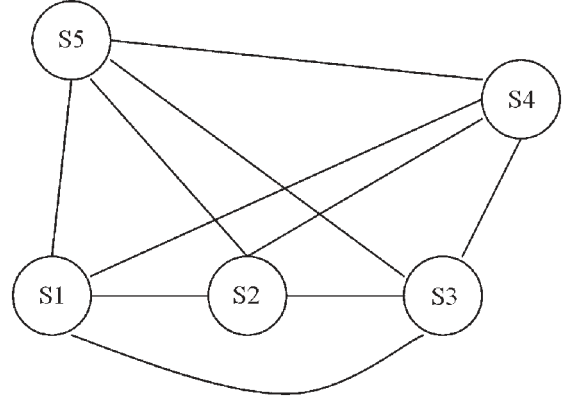


Fig. 2. Example of synergy graph with five sensors.

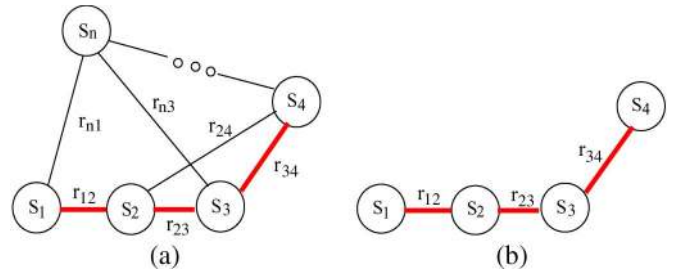


Fig. 3. (a) Synergy chain $\{S_1, S_2, S_3, S_4\}$ (highlighted) on a pruned synergy graph. (b) Corresponding MSC.

We use the synergy graph to graphically represent the synergy among multiple sensors. By definition of the synergy, G is a complete graph, i.e., there is a link between any two nodes in the graph. Fig. 2 gives an example synergy graph consisting of five sensors.

Definition 4 (Pruned Synergy Graph): A pruned synergy graph is created from a synergy graph after removing some links. A pruned synergy is, therefore, not a complete graph.

Fig. 3 shows an example of a pruned synergy graph. To further exploit the theoretical properties of mutual information $I(\Theta; \mathbf{S})$ for a set of sensors, we give the following definitions.

Definition 5 (Synergy Chain): Given a pruned synergy graph G , if all the sensors in a subset on G are serially linked, then this subset of sensors is referred to as a sensor synergy chain. Note that while the sensors in a set \mathbf{S} are generally order independent, the sensors in a synergy chain are order dependent and sequentially ordered.

Definition 6 (MSC): Given a synergy chain with n sensors, for all $i = 1, \dots, n - 1$, if $p(S_{i+1} | S_1, S_2, \dots, S_i) = p(S_{i+1} | S_i)$, then the chain that describes the synergetic relationship among $\{S_1, \dots, S_n\}$ is a Markov synergy chain (MSC). An MSC is also ordered.

Fig. 3 graphically shows the above definitions about the synergy chain in a pruned synergy graph. The MSC represents an ideal synergetic relationship among sensors. The MSC rarely exists in practice, but this does not prevent us from using it as a basis for the graph-theoretic analysis of synergy among sensors. In fact, as to be shown later, the concept of MSC is used to define the upper bound for the mutual information of a set of sensors. With the above definitions, we give the following two theorems.

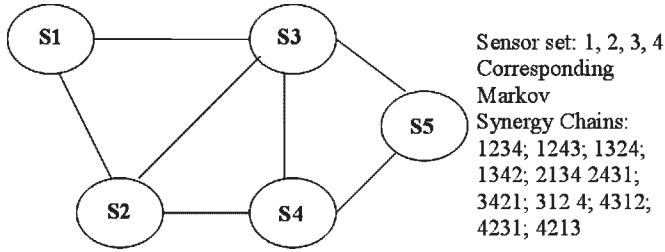


Fig. 4. Illustration of a set of possible MSCs for a set of four sensors in a pruned synergy graph.

Theorem 1 (MSC Rule): Given an MSC with a set of ordered sensors $\mathbf{S} = \{S_1, \dots, S_n\}$, for any n , the joint mutual information with respect to Θ for sensors on an MSC is

$$I^M(\Theta; S_1, \dots, S_n) = I(\Theta; S_1) + \sum_{i=1}^{n-1} (I(\Theta; S_i, S_{i+1}) - I(\Theta; S_i)). \quad (7)$$

The proof of this theorem can be found in Appendix A. We want to make note that the mutual information for an MSC is sensor-order dependent due to the pairwise synergy definition. The significance of Theorem 1 is that it allows us to efficiently compute the joint mutual information for n ($n > 2$) ordered sensors as a sum of mutual information of only singleton and pairwise sensors if the set of sensors forms an MSC. In contrast to (4), the computational cost of (7) is dramatically reduced. Although (7) is particularly for an MSC, the theorem above has some useful properties that can be used for the solution of our sensor selection problem.

Theorem 2 (Synergy Upper Bound): For a set of unordered sensors $\mathbf{S} = \{S_1, \dots, S_n\}$, its mutual information is upper bounded by the mutual information of the corresponding MSC, i.e.,

$$I(\Theta; S_1, \dots, S_n) \leq I^M(\Theta; S_1, \dots, S_n). \quad (8)$$

The proof of this theorem is provided in Appendix B. Please note that while $I(\Theta; S_1, \dots, S_n)$ is sensor-order independent, $I^M(\Theta; S_1, \dots, S_n)$ is sensor-order dependent. As a result, depending on the order of sensors in \mathbf{S} , different MSCs may be produced. Let

$$\begin{aligned} I_{\min}^M &= \arg \min_{\mathcal{S}} (I^M(\Theta; \mathcal{S})) \\ I_{\max}^M &= \arg \max_{\mathcal{S}} (I^M(\Theta; \mathcal{S})) \end{aligned} \quad (9)$$

where \mathcal{S} denotes all the possible orders of a sensor set $\{S_1, \dots, S_n\}$. I_{\min}^M is referred to as the LUB of $I(\Theta; S_1, \dots, S_n)$, and I_{\max}^M is referred to as the greatest upper bound (GUB) of $I(\Theta; S_1, \dots, S_n)$. For example, in Fig. 4, the sensor set $\mathbf{S} = \{S_1, S_2, S_3, S_4\}$ has multiple MSCs, as given in this figure, and there exist a LUB and a GUB of $I(\Theta; \mathbf{S})$.

We are particularly interested in the LUB of $I(\Theta; \mathbf{S})$ due to two reasons. First, it can be seen from Figs. 5 and 6 that the LUBs of $I(\Theta; \mathbf{S})$ closely follow the trend of $I(\Theta; \mathbf{S})$ in the entire space of sensor subsets. Second, the exact value of $I(\Theta; \mathbf{S})$ and its LUB are quantitatively very close in value. Thus, $I_{\min}^M(\Theta; \mathbf{S})$ provides a substitute measure for $I(\Theta; \mathbf{S})$

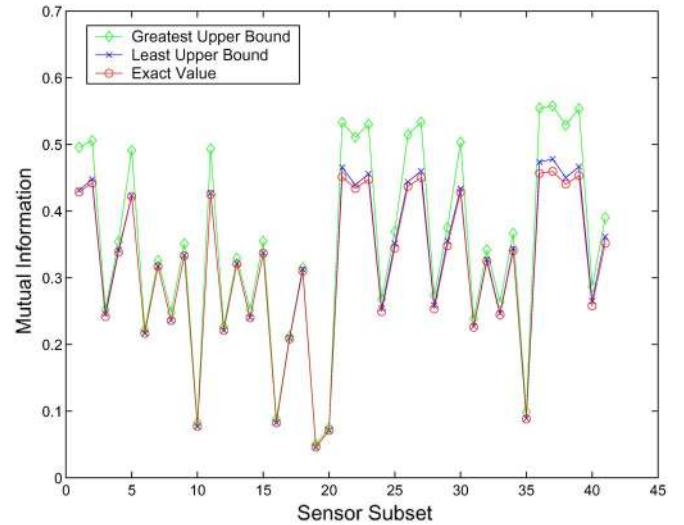


Fig. 5. Bound of mutual information $I(\Theta, S)$ and its exact value from a six-sensor BN model. The X -axis represents the indexes of 41 sensor subsets. Labels 1–20 are the indexes of the three-sensor subsets; Labels 21–35 are the indexes of the four-sensor subsets; and Labels 36–41 are the indexes of the five-sensor subsets.

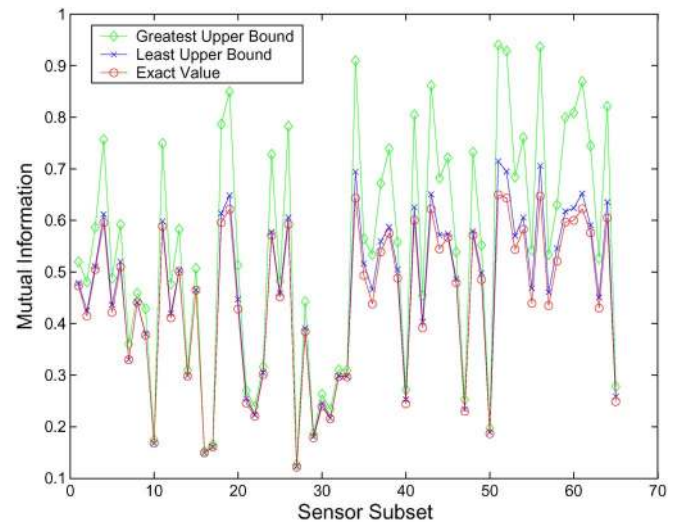


Fig. 6. Bound of mutual information $I(\Theta, S)$ and its exact value from a ten-sensor BN model. The X -axis represents the indexes of sensor subsets. For clarity, the figure only shows 66 subsets out of 627. Labels 1–18 are the indexes of the five-sensor subsets; Labels 19–34 are the indexes of the six-sensor subsets; Labels 35–51 are the indexes of the seven-sensor subsets; and Labels 52–66 are the indexes of the eight-sensor subsets.

that can be used to evaluate an optimal sensor subset. Importantly, the LUBs of $I(\Theta; \mathbf{S})$ can simply be written as the sum of the mutual information of only pairwise sensors and singleton sensors, as shown in (7), hence, with relatively very low computational cost. Therefore, the computational difficulty in exactly computing the higher-order mutual information can be circumvented by only computing the LUBs of the mutual information. This is the central strategy of our approach.

B. Pruning Synergy Graph

The synergy graph is a completely connected network due to the weights of synergy graph $r_{ij} \geq 0$. Some sensors are highly synergistic, whereas others are not. Intuitively, sensors that

TABLE I
EXAMPLE OF SYNERGY COEFFICIENT WITHOUT PRUNING

r_{ij}	1	2	3	4	5	6	7	8	9	10
1	0	0.0004	0.0034	0.0034	0.0029	0.0023	0.0035	0.0035	0.0031	0.0017
2		0	0.0004	0.0004	0.0004	0.0003	0.0004	0.0004	0.0004	0.0002
3			0	0.0215	0.0196	0.0156	0.0099	0.0122	0.0212	0.0113
4				0	0.0184	0.0147	0.0099	0.0122	0.0198	0.0105
5					0	0.0930	0.0083	0.0104	0.0650	0.0659
6						0	0.0066	0.0082	0.0517	0.1329
7							0	0.0100	0.0091	0.0050
8								0	0.0112	0.0060
9									0	0.0388
10										0

cause a very small reduction in uncertainty of hypotheses are those that give us the least additional information beyond what we would obtain from other sensors. In such cases, r_{ij} is very small. We prune the sensor synergy graph so that many weak sensor combinations are eliminated while preserving the most promising ones. This can significantly reduce the search space in identifying the optimal sensor subset. We prune the synergy matrix (the weights of the synergy graph) in (6) by using

$$r_{ij} = \begin{cases} 1, & r_{ij} > \tau \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where τ is a pruning threshold. The selection of an appropriate threshold τ is problem dependent. We want to note that although there is no theoretical basis to determine a good pruning threshold, our empirical tests, however, show that using the arithmetic average of r_{ij} as the pruning threshold can preserve most of the strong synergetic connections in the graph while eliminating weak links. After pruning, a fully connected synergy graph then becomes a sparse graph. Table I and

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ & & & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ & & & & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ & & & & & 0 & 0 & 0 & 1 & 1 & 0 \\ & & & & & & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & 0 & 0 & 0 & 0 \\ & & & & & & & & 0 & 1 & 0 \\ & & & & & & & & & & 0 \end{bmatrix} \quad (11)$$

are examples of the synergy coefficient before and after pruning. Fig. 7 illustrates their corresponding synergy graph from a completely connected network to a sparse graph after pruning.

C. Computing Pairwise Mutual Information

In the above sections, we assumed that we have known the mutual information of the pairwise sensors $I(\Theta; S_i, S_j)$. For n sensors, there are $(n(n - 1)/2)$ pairs of sensors. To obtain the mutual information for one pair of sensors, it requires four repetitions of inferences if the sensor state is binary. Therefore, $2n(n - 1)$ repetitions of inference are needed for

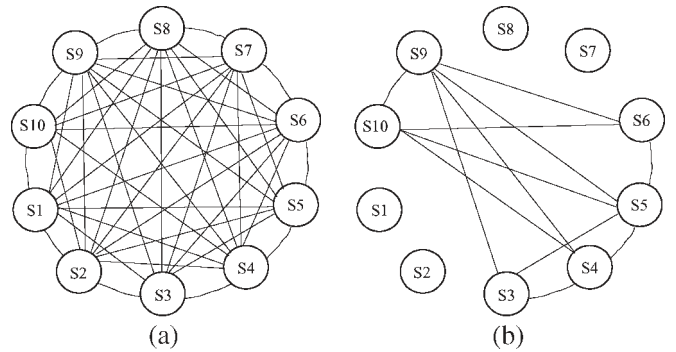


Fig. 7. (a) Completely connected synergy graph and the links are weighted by r_{ij} , as shown in Table I. (b) Pruned synergy graph and its corresponding matrix as shown in (11). The pruning threshold is the average of r_{ij} , and it is 0.0161.

all pairs of sensors. Although this computation is manageable, it still severely limits the performance as n becomes large. Fortunately, there is an efficient way to compute the mutual information for all pairs of sensors [15], [16].

Referring to Fig. 1, the joint probability of hypothesis Θ and pairwise sensors $\{S_i, S_j\}$ may be written as in (12), shown at the bottom of the next page, where $\pi(x)$ represents the parental nodes of node x . From (12), it can be observed that the first factor $P(\Theta) \prod_{k=1}^K P(X_k | \pi(X_k)) \prod_{m=1}^M P(Y_m | \pi(Y_m))$ is related to the part of the BN structure that does not include the sensors. The structure is, therefore, fixed, and so are its probabilities. Hence, this term is constant, independent of the pair of sensors used. On the other hand, the second factor $\{\sum_{S_1, S_2, \dots, S_N, l \neq i, l \neq j} \prod_{n=1}^N P(S_n | \pi(S_n))\}$ varies, depending on the pair of sensors selected. Therefore, we do not need to recalculate the unchanged part (the first factor) of (12) at each time. Instead, we only need to compute it once for all pairs of sensors, but use it over time so that the computation of pairwise mutual information can significantly be curtailed. Given $P(\Theta, S_i, S_j)$, it can then be substituted into (4) to compute $I(\Theta; S_i, S_j)$. Details of this method can be found in [15] and [16]. Fig. 8 illustrates the comparative result of time saving in computing (12) for all pairs of sensors by using our method and by directly using two inference algorithms, namely, clique tree propagation (CTP) [17], [18] and variable elimination (VE) [19]. The evaluation is performed on a six-layer BN model with 10, 15, and 20 sensors.

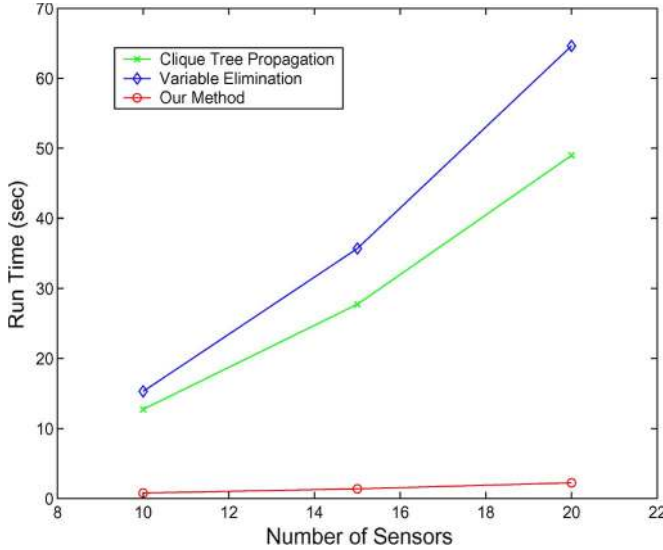


Fig. 8. Comparison of time saving among CTP, VE, and our method in computing (12) for all pairs of sensors. It can be seen that our method can significantly save time.

D. Approximation Algorithm

We are now ready to provide the complete algorithm. Let \mathbf{S} denote the current set of selected sensors, and let $\text{lub}(\Theta; \mathbf{S})$ be the LUB of $I(\Theta; \mathbf{S})$. The approximation sensor selection algorithm is given in Table II. Guided by the pruned synergy graph, the algorithm starts with the best pair of sensors identified through an exhaustive search and then searches for the next best sensor. The next best sensor is the one, when added to the current sensor ensemble, that yields the highest utility, which is computed from $\text{lub}(\Theta; \mathbf{S})$. This process repeats, with one sensor added to the current sensor ensemble at a time, until the newly added sensor does not yield an improvement in sensor utility. Although the algorithm is greedy, the searching process is guided by a synergy graph so that the selected sensor subset is serially connected. This, therefore, ensures both the quality and the speed of sensor selection.

IV. ALGORITHM EVALUATION

Since the main contribution of this paper is the introduction of an alternative measure to mutual information for efficient sensor selection, the experimental evaluation should focus on the effectiveness of this measure for both sensor selection accuracy and efficiency. We want to emphasize that the alternative measure, i.e., the LUB of mutual information, is an approximation of the mutual information only for the purpose of sensor selection. As a result, the quality of this approximation should

TABLE II
PSEUDOCODE OF THE APPROXIMATION ALGORITHM
TO SELECT A SUBSET OF SENSORS

SENSOR-SELECTION(n)

- 1 **for** each i, j , compute $I(\Theta; S_i)$ and $I(\Theta; S_i, S_j)$
- 2 Construct a pruned synergy graph G
- 3 Choose S_{i^*}, S_{j^*} such that $I(\Theta; S_i, S_j)$ is maximized for all i and j
- 4 $\mathbf{S} \leftarrow \{S_{i^*}, S_{j^*}\}$
- 5 **while** $|\mathbf{S}| < m$
- 6 **for** each \mathbf{S}' , where $|\mathbf{S}'| = |\mathbf{S}| + 1$, and \mathbf{S}' is a synergy chain on G and $\mathbf{S} \subset \mathbf{S}'$
- 7 Find all Markov synergy chains of \mathbf{S}'
- 9 $\text{lub}(\Theta; \mathbf{S}') \leftarrow \arg \min I(\Theta; \mathbf{S}')$,
where $I(\Theta; \mathbf{S}')$ is computed by Eq. (7), and min is taken over all Markov synergy chains of \mathbf{S}'
- 10 $\mathbf{S}^{**} \leftarrow \arg \max \text{lub}(\Theta; \mathbf{S}')$
where max is taken over all \mathbf{S}'
- 11 **if** $\text{lub}(\Theta; \mathbf{S}^{**}) > \text{lub}(\Theta; \mathbf{S})$
- 12 $\mathbf{S} \leftarrow \mathbf{S}^{**}$
- 13 **else break**
- 14 **return** \mathbf{S}

be evaluated against its performance in sensor selection. For this, we propose to measure how close the sensor selection results using the alternative measure are to those based on mutual information. The closeness between a sensor subset selected using the alternative measure and a sensor subset selected based on mutual information is quantified by the relative difference in mutual information. Based on this criterion, we will experimentally evaluate the proposed method under different BN topologies, different BN model complexities, and different number of sensors.

Given two different criteria (mutual information and its LUB) for measuring sensor gain, sensor selection can be carried out by using different methods. We will perform sensor selection using the following methods: 1) brute-force method; 2) random method, which randomly chooses one sensor at a time to form a sensor ensemble; and 3) the proposed method. These experiments try to demonstrate the following: 1) The proposed LUB criterion suboptimally works for different methods. 2) Given the same sensor selection criterion, the proposed greedy approach outperforms the random sensor selection method.

$$\begin{aligned}
 P(\Theta, S_i, S_j) &= \sum_{S_1, S_1, \dots, S_n, l \neq i, l \neq j} \left\{ P(\Theta) \prod_{k=1}^K P(X_k | \pi(X_k)) \prod_{m=1}^M P(Y_m | \pi(Y_m)) \prod_{n=1}^N P(S_n | \pi(S_n)) \right\} \\
 &= P(\Theta) \prod_{k=1}^K P(X_k | \pi(X_k)) \prod_{m=1}^M P(Y_m | \pi(Y_m)) \left\{ \sum_{S_1, S_1, \dots, S_n, l \neq i, l \neq j} \prod_{n=1}^N P(S_n | \pi(S_n)) \right\} \quad (12)
 \end{aligned}$$

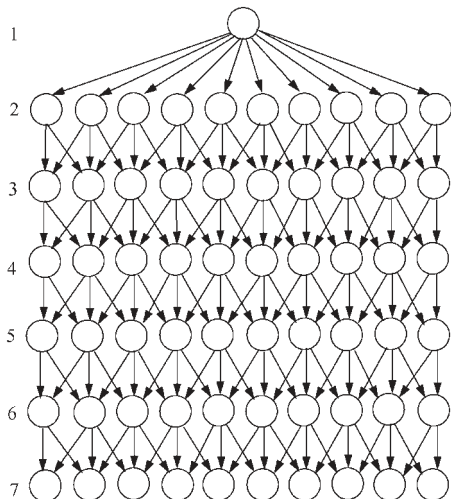


Fig. 9. Generic example of the BN network used for evaluation, where the top layer is for the hypothesis, and the bottom layer is for the sensors. The intermediate layers are arbitrarily and randomly connected.

We first compare the performance of the proposed sensor selection method in Table II with the brute-force method. The brute-force method exhaustively identifies the best sensor subset by the exact mutual information. The study is done by using different numbers of sensors and different BN topologies. Fig. 9 shows a generic example of a BN used for the evaluation.

Due to the exponential time with the brute-force approach, we limit our test models to up to five layers and up to ten sensors or less. The exact number of layers, the connections among nodes in the intermediate layers, and the number of sensors are randomly generated so that ten different BNs with different topologies are generated. For each randomly generated BN topology, its parameters are randomly parameterized ten times to produce ten differently parameterized BNs for each selected topology. This yields a total of 100 test models. Fig. 10 shows two examples of BNs used for this paper.

The results averaged among 100 trials are shown in Table III, where the closeness is defined as the relative difference in mutual information between the solution from our approach and the solution from the brute-force approach. It can be seen that the solution with our method is close to the sensor selection results using the brute-force method. For further comparison, the run time of the two methods is measured on a 2.0-GHz computer, and the run time averaged among ten trials is summarized in Table III. Compared with the brute-force method, our method significantly reduces the computation time with minimum loss in sensor selection accuracy.

To demonstrate the improvement of the proposed method over random sensor selection, the results of random sensor selection are also included in Table III. For a fair comparison, we first use our method to select a best subset and then use the random method to select a subset of the same size using the same criterion. To account for the random nature of random selection, the results are averaged, and the averaged result is used to compare against the result from our method. Compared with the random sensor selection, our method shows a significant improvement in sensor selection accuracy.

Finally, we want to note that the randomly generated BN topologies (for example, the BNs in Figs. 9 and 10) may not

necessarily satisfy the assumption needed for Theorem 1 to hold. Despite this, the selected sensors remain close (in mutual information) to those selected by the brute-force method, as demonstrated in Table III. We also repeat the above experiments by using naive BNs. The parameters of BNs are randomly generated. We selected k sensors from n sensors ($k < n$) without considering sensor costs. The sensors selected by the brute-force method and by our approach have no difference.

V. CONCLUSION

It is computationally difficult to identify an optimal sensor subset with the information-theoretic criterion. To address problem, we have presented an approximation method to find a near-optimal sensor subset by utilizing the sensor pairwise information to infer the synergy among sensors. Specifically, this paper includes the following aspects: First, we propose to use a BN to represent sensors, their dependencies, and their relationships to other latent variables. In addition, the built-in conditional independence assumptions with the BNs allow factorizing the joint probabilities so that fusion can efficiently be performed. Second, we introduce a statistical measure to quantify the pairwise synergy among sensors. Based on the synergy measure, a synergy graph is constructed, which is used to infer synergy among multiple sensors, based on which we can then eliminate many unpromising sensor combinations. Finally, for the remaining sensor combinations, a greedy approach is introduced to identify the optimal sensor combination based on the LUB of the joint mutual information. The use of the LUB of the joint mutual information instead of the joint mutual information itself significantly reduces the computation time with minimum loss in accuracy. We demonstrate both the optimality and the efficiency of the proposed method through many random simulations under different numbers of sensors and different relationships among sensors.

A major assumption of this paper is that the two sensors are conditionally independent of each other, given another sensor between the two sensors and the fusion result. This assumption could limit the utility of this paper. As part of the future research, we will study ways to overcome this assumption. Another assumption we made in this paper is that all the sensors have the same cost. Such an assumption is not realistic for many applications. Overcoming this assumption, however, requires incorporating the sensor cost into the proposed synergy function, which is a nontrivial task. We will study this issue in the future as well.

APPENDIX

In the following, we introduce our proof for Theorems 1 and 2.

A. Proof of Theorem 1

Before proving Theorem 1, we give the following lemma.

Lemma 1 (Chain Rule of Mutual Information): Letting X, Y_1, \dots, Y_m be random variables, then

$$I(X; Y_1, \dots, Y_m) = I(X; Y_1) + \sum_{i=2}^m I(X; Y_i | Y_1, \dots, Y_{i-1}). \quad (13)$$

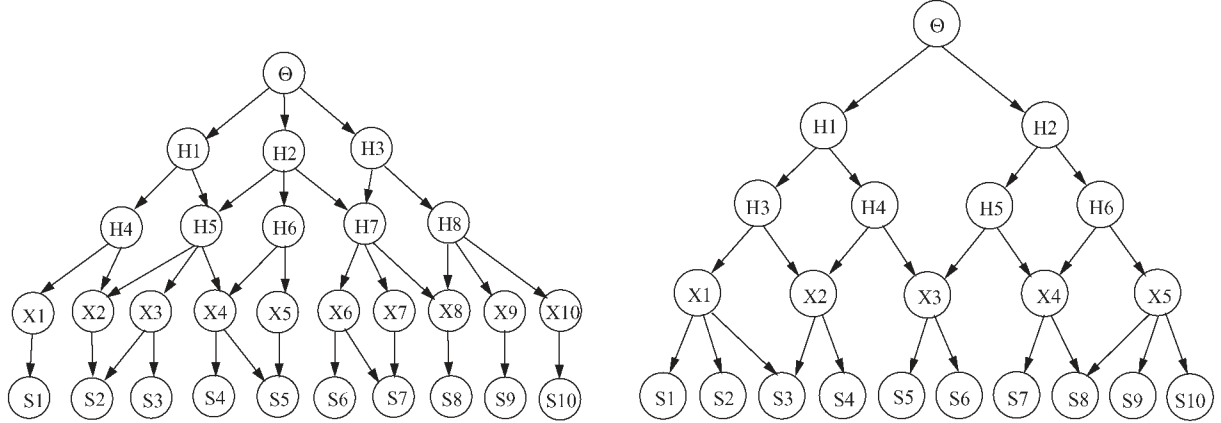


Fig. 10. Two specific examples of BN structures with different numbers of sensors used for the evaluation.

TABLE III
COMPARISON OF THE PROPOSED METHOD AND THE BRUTE-FORCE METHOD

Number of Sensors	Our Approach		Random Method	Brute-Force
	Relative mutual information difference of our method to brute force methods	Run time (Seconds)	Relative mutual information difference of random method to brute force methods	Run time (Seconds)
7	1.56%	1.020	21.13%	63.87
8	1.77%	1.099	28.32%	355.05
9	2.75%	1.209	36.54%	2967.36
10	1.89%	1.430	39.19%	13560.54

The proof of Lemma 1 is straightforward [2]. We now turn to proving Theorem 1.

Proof: Based on Lemma 1, we have

$$\begin{aligned}
 & I(\Theta; S_1, \dots, S_m) \\
 &= I(\Theta; S_1) + I(\Theta; S_2 | S_1) + I(\Theta; S_3 | S_1, S_2) \\
 & \quad + I(\Theta; S_4 | S_1, S_2, S_3) + \dots + I(\Theta; S_m | S_1, \dots, S_{m-1}).
 \end{aligned} \tag{14}$$

We start with an MSC containing four random variables $\{\Theta, S_1, S_2, S_3\}$, then extend it to five variables, and finally to a finite number of arbitrary random variables forming an MSC. Notice that Θ is the hypothesis, and S_1, S_2, S_3 are the sensors.

Based on Definition 6 of MSC, S_1 and S_3 are conditionally independent given S_2 , i.e., $P(S_3 | S_1, S_2) = P(S_3 | S_2)$. First, we prove that the following equation holds when S_1 and S_3 are conditionally independent given S_2 :

$$\begin{aligned}
 & I(\Theta; S_3 | S_2) = I(\Theta; S_3 | S_1, S_2) \\
 & I(\Theta; S_3 | S_1, S_2)
 \end{aligned} \tag{15}$$

$$\begin{aligned}
 &= \sum_{\Theta, S_1, S_2, S_3} p(\theta, s_1, s_2, s_3) \left\{ \lg \frac{p(\theta, s_3 | s_1, s_2)}{p(\theta | s_1, s_2)p(s_3 | s_1, s_2)} \right\} \\
 &= \sum_{\Theta, S_1, S_2, S_3} p(\theta, s_1, s_2, s_3) \left\{ \lg \frac{p(\theta, s_3, s_1, s_2)}{p(\theta, s_1, s_2)p(s_3 | s_2)} \right\} \\
 &= \sum_{\Theta, S_1, S_2, S_3} p(\theta, s_1, s_2, s_3) \left\{ \lg \frac{p(s_3 | \theta, s_1, s_2)}{p(s_3 | s_2)} \right\}
 \end{aligned}$$

$$\begin{aligned}
 &= \sum_{\Theta, S_1, S_2, S_3} p(\theta, s_1, s_2, s_3) \left\{ \lg \frac{p(s_3 | \theta, s_2)}{p(s_3 | s_2)} \right\} \\
 &= \sum_{\Theta, S_2, S_3} p(\theta, s_2, s_3) \left\{ \lg \frac{p(s_3, \theta | s_2)}{p(\theta | s_2)p(s_3 | s_2)} \right\} \\
 &= I(\Theta; S_3 | S_2).
 \end{aligned} \tag{16}$$

Please note that for the derivations in (16), we assume that $p(S_3 | \Theta, S_1, S_2) = p(S_3 | \Theta, S_2)$, i.e., S_3 and S_1 are conditionally independent given both Θ and S_2 , where Θ is a random variable representing the fusion result, and S_i is a sensor. The typical relationships between Θ and S_i 's are illustrated in Fig. 1, where Θ is typically the root node, and S_i 's are the leaf nodes in the BN. Given this understanding, if the BN is such that the path (undirected path) between two sensor nodes (e.g., S_1 and S_3) goes through Θ node (e.g., the BN in Fig. 1), then following the D-separation principle for BN, $p(S_3 | \Theta, S_1, S_2) = p(S_3 | \Theta, S_2)$ holds. Please note that this assumption only holds for some BNs, such as the one in Fig. 1 and the naive BN. It may not hold for an arbitrary BN.

From the chain rule of mutual information, we have

$$I(\Theta; S_3, S_2) = I(\Theta; S_2) + I(\Theta; S_3 | S_2). \tag{17}$$

Hence, combining (16) and (17) yields

$$I(\Theta; S_3 | S_1, S_2) = I(\Theta; S_2, S_3) - I(\Theta; S_2). \tag{18}$$

Now, we want to apply the similar algebraic process to prove $I(\Theta; S_4 | S_1, S_2, S_3) = I(\Theta; S_4 | S_3)$ in (19), shown at the bottom of the page, given the Markov conditions that $P(S_4 | S_2, S_3) = P(S_4 | S_3)$, $P(S_1 | S_3, S_2) = P(S_1 | S_2)$, $P(S_4 | S_1, S_2) = P(S_4 | S_2)$, and $P(S_4 | S_1, S_3) = P(S_4 | S_3)$. By mutual information chain rule, we have $I(\Theta; S_3, S_4) = I(\Theta; S_3) + I(\Theta; S_4 | S_3)$, i.e.,

$$I(\Theta; S_4 | S_3) = I(\Theta; S_3, S_4) - I(\Theta; S_3). \quad (20)$$

Combining (19) and (20) produces

$$\begin{aligned} I(\Theta; S_4 | S_1, S_2, S_3) &= I(\Theta; S_4 | S_3) \\ &= I(\Theta; S_3, S_4) - I(\Theta; S_3). \end{aligned} \quad (21)$$

Finally, we can generalize the above process to prove

$$\begin{aligned} I(\Theta, S_m | S_1, S_2, \dots, S_{m-1}) \\ = I(\Theta; S_{m-1}, S_m) - I(\Theta; S_{m-1}). \end{aligned} \quad (22)$$

Substituting the results in (18), (21), and (22) into (14) yields

$$\begin{aligned} I(\Theta; S_1, S_2, \dots, S_m) \\ &= I(\Theta; S_1) + I(\Theta; S_2 | S_1) + I(\Theta; S_3, S_2) \\ &\quad - I(\Theta; S_2) + I(\Theta; S_3, S_4) - I(\Theta; S_3) + \dots \\ &\quad + I(\Theta; S_{m-1}, S_m) - I(\Theta; S_{m-1}) \\ &= I(\Theta; S_1) + I(\Theta; S_2, S_1) - I(\Theta; S_1) + I(\Theta; S_3, S_2) \\ &\quad - I(\Theta; S_2) + I(\Theta; S_3, S_4) - I(\Theta; S_3) + \dots \\ &\quad + I(\Theta; S_{m-1}, S_m) - I(\Theta; S_{m-1}) \\ &= I(\Theta; S_1) + \sum_{i=2}^M I(\Theta; S_{m-1}, S_m) - I(\Theta; S_{m-1}). \end{aligned} \quad (23)$$

This completes the proof for Theorem 1. \blacksquare

B. Proof of Theorem 2

Proof: We want to prove

$$I(\Theta; S_1, \dots, S_m) \leq I^M(\Theta; S_1, \dots, S_m). \quad (24)$$

From the mutual information chain rule, we have

$$\begin{aligned} I(\Theta; S_1, S_2, \dots, S_m) \\ &= I(\Theta; S_1) + I(\Theta; S_2 | S_1) \\ &\quad + I(\Theta; S_3 | S_1, S_2) + I(\Theta; S_4 | S_1, S_2, S_3) + \dots \\ &\quad + I(\Theta; S_m | S_1, S_2, \dots, S_{m-1}). \end{aligned} \quad (25)$$

By Theorem 1, we have

$$\begin{aligned} I^M(\Theta; S_1, \dots, S_m) \\ &= I(\Theta; S_1) + I(\Theta; S_2 | S_1) + I(\Theta; S_3 | S_2) \\ &\quad + I(\Theta; S_4 | S_3) + \dots + I(\Theta, S_m | S_{m-1}). \end{aligned} \quad (26)$$

By the definition of mutual information, we have

$$I(\Theta; S_3; S_2; S_1) = I(\Theta; S_3; S_2) - I(\Theta; S_3; S_2 | S_1) \quad (27)$$

which readily leads to

$$I(\Theta; S_3; S_2 | S_1) = I(\Theta; S_3 | S_2) - I(\Theta; S_3 | S_2, S_1). \quad (28)$$

Hence

$$I(\Theta; S_3 | S_2, S_1) = I(\Theta; S_3 | S_2) - I(\Theta; S_3; S_2 | S_1). \quad (29)$$

Therefore

$$I(\Theta; S_3 | S_2) \geq I(\Theta; S_3 | S_1, S_2).$$

Please note that we assume here that $I(\Theta; S_3; S_2 | S_1) > 0$, which is correct since for our application Θ (the hypothesis)

$$\begin{aligned} I(\Theta; S_4 | S_1, S_2, S_3) \\ &= \sum_{\Theta, S_1, S_2, S_3, S_4} p(\theta, s_1, s_2, s_3, s_4) \left\{ \lg \frac{p(\theta, s_4 | s_1, s_2, s_3)}{p(\theta | s_1, s_2, s_3)p(s_4 | s_1, s_2, s_3)} \right\} \\ &= \sum_{\Theta, S_1, S_2, S_3, S_4} p(\theta, s_1, s_2, s_3, s_4) \left\{ \lg \frac{p(\theta, s_4, s_1, s_2, s_3)}{p(\theta, s_1, s_2, s_3)p(s_4 | s_3)} \right\} \\ &= \sum_{\Theta, S_1, S_2, S_3, S_4} p(\theta, s_1, s_2, s_3, s_4) \left\{ \lg \frac{p(s_4 | \theta, s_1, s_2, s_3)}{p(s_4 | s_3)} \right\} \\ &= \sum_{\Theta, S_1, S_2, S_3, S_4} p(\theta, s_1, s_2, s_3, s_4) \left\{ \lg \frac{p(s_4 | \theta, s_3)}{p(s_4 | s_3)} \right\} \\ &= \sum_{\Theta, S_3, S_4} p(\theta, s_3, s_4) \left\{ \lg \frac{p(s_4 | \theta, s_3)}{p(s_4 | s_3)} \right\} \\ &= \sum_{\Theta, S_2, S_3} p(\theta, s_3, s_4) \left\{ \lg \frac{p(s_4, \theta | s_3)}{p(\theta | s_3)p(s_4 | s_3)} \right\} = I(S_4; \Theta | S_3) = I(\Theta; S_4 | S_3) \end{aligned} \quad (19)$$

and the other variables (sensors) are not independent of each other.

Similarly, we have $I(\Theta; S_4 | S_3) \geq I(\Theta; S_4 | S_1, S_2, S_3)$ and $I(\Theta; S_m | S_{m-1}) \geq I(S_m | S_1, S_2, \dots, S_{m-1})$.

Hence, (26) \geq (25). The equality sign holds when the Markov property between neighbor sensors is true.

Hence, this completes the proof for Theorem 2. ■

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable and constructive comments, which help to significantly improve this paper.

REFERENCES

- [1] Y. Zhang and Q. Ji, "Active and dynamic information fusion for multisensor systems with dynamic Bayesian networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 2, pp. 467–472, Apr. 2006.
- [2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
- [3] D. E. Heckerman, E. Horvitz, and B. N. Nathwani, "Toward normative expert systems: Part I. The pathfinder project," *Methods Inf. Med.*, vol. 31, no. 2, pp. 90–105, Jun. 1992.
- [4] L. van der Gaag and M. Wessels, "Selective evidence gathering for diagnostic belief networks," *AISB Q.*, vol. 86, pp. 23–34, 1993.
- [5] S. Dittmer and F. Jensen, "Myopic value of information in influence diagrams," in *Uncertainty Artif. Intell.*, 1997, pp. 142–149.
- [6] V. Bayer-Zubek, "Learning diagnostic policies from examples by systematic search," in *Uncertainty Artif. Intell.*, 2004, pp. 27–34.
- [7] V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden Markov model sensors," *IEEE Trans. Signal Process.*, vol. 50, no. 6, pp. 1382–1397, Jun. 2002.
- [8] E. Ertin, J. W. Fisher, and L. C. Potter, "Maximum mutual information principle for dynamic sensor query problems," in *Proc. Inf. Process. Sens. Netw.*, San Francisco, CA, 2003, pp. 405–416.
- [9] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based sensor selection heuristic for target localization," in *Proc. Inf. Process. Sens. Netw.*, Berkeley, CA, 2004, pp. 36–45.
- [10] D. Heckerman, E. Horvitz, and B. Middleton, "An approximate nonmyopic computation for value of information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 3, pp. 292–298, Mar. 1993.
- [11] A. Krause and C. Guestrin, "Optimal nonmyopic value of information in graphical models—Efficient algorithms and theoretical limits," in *Int. Joint Conf. Artif. Intell.*, 2005, pp. 1339–1345.
- [12] A. Krause and C. Guestrin, "Near-optimal nonmyopic value of information in graphical models," in *Uncertainty Artif. Intell.*, 2005, pp. 324–333.
- [13] W. Liao and Q. Ji, "Efficient active fusion for decision-making via variational approximation," in *21th Nat. Conf. Artif. Intell.*, 2006, pp. 1180–1185.
- [14] Y. Zhang and Q. Ji, "Sensor selection for active information fusion," in *20th Nat. Conf. Artif. Intell.*, 2005, pp. 1229–1234.
- [15] W. Liao, W. Zhang, and Q. Ji, "A factor tree inference algorithm for Bayesian networks and its application," in *Proc. 16th ICTAI*, Boca Raton, FL, 2004, pp. 652–656.
- [16] W. Zhang and Q. Ji, "A factorization approach to evaluating simultaneous influence diagrams," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 4, pp. 746–757, Jul. 2006.
- [17] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.
- [18] F. V. Jensen and F. Jensen, "Optimal junction trees," in *Proc. 10th Conf. Uncertainty AI*, San Francisco, CA, 1994, pp. 360–366.
- [19] R. Dechter, "Bucket elimination: A unifying framework for probabilistic inference," in *Proc. 12th Conf. Uncertainty AI*, San Francisco, CA, 1996, pp. 211–219.



Yongmian Zhang (M'04) received the Ph.D. degree in computer engineering from the University of Nevada, Reno, in 2004.

He holds a research position with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. His areas of research include information fusion, computer vision, and human-computer interactions.



Qiang Ji (SM'04) received the Ph.D. degree in electrical engineering from the University of Washington, Seattle.

He is currently a Professor with the Department of Electrical, Computer, and Systems Engineering and the Director of the Intelligent Systems Laboratory, Rensselaer Polytechnic Institute, Troy, NY. He is also a Program Director with the National Science Foundation (NSF), Arlington, VA, where he is responsible for managing part of the NSF's computer vision and machine learning programs. He previously held teaching and research positions with Beckman Institute, University of Illinois, Urbana; Robotics Institute, Carnegie Mellon University, Pittsburgh, PA; the Department of Computer Science, University of Nevada, Reno; and the U.S. Air Force Research Laboratory. His research has been supported by major governmental agencies, including the NSF, the National Institutes of Health, the Defense Advanced Research Projects Agency, the Office of Naval Research, the Army Research Office, and the Air Force Office of Scientific Research, as well as by major companies, including Honda and Boeing. He has published more than 150 papers in peer-reviewed journals and conference proceedings. He is an Editor for several computer-vision- and pattern-recognition-related journals.

Prof. Ji has served as program committee member, area chair, and program chair in numerous international conferences/workshops.

Approximate Nonmyopic Sensor Selection via Submodularity and Partitioning

Wenhui Liao, Qiang Ji, *Senior Member, IEEE*, and William A. Wallace, *Fellow, IEEE*

Abstract—As sensors become more complex and prevalent, they present their own issues of cost effectiveness and timeliness. It becomes increasingly important to select sensor sets that provide the most information at the least cost and in the most timely and efficient manner. Two typical sensor selection problems appear in a wide range of applications. The first type involves selecting a sensor set that provides the maximum information gain within a budget limit. The other type involves selecting a sensor set that optimizes the tradeoff between information gain and cost. Unfortunately, both require extensive computations due to the exponential search space of sensor subsets. This paper proposes efficient sensor selection algorithms for solving both of these sensor selection problems. The relationships between the sensors and the hypotheses that the sensors aim to assess are modeled with Bayesian networks, and the information gain (benefit) of the sensors with respect to the hypotheses is evaluated by mutual information. We first prove that mutual information is a submodular function in a relaxed condition, which provides theoretical support for the proposed algorithms. For the budget-limit case, we introduce a greedy algorithm that has a constant factor of $(1 - 1/e)$ guarantee to the optimal performance. A partitioning procedure is proposed to improve the computational efficiency of the algorithms by efficiently computing mutual information as well as reducing the search space. For the optimal-tradeoff case, a submodular–supermodular procedure is exploited in the proposed algorithm to choose the sensor set that achieves the optimal tradeoff between the benefit and cost in a polynomial-time complexity.

Index Terms—Active fusion, Bayesian networks (BNs), sensor selection, submodular function.

I. INTRODUCTION

MANY real-world applications use sensors to obtain information that will help them improve their activities. Here, sensor is a general term; it could refer to a test, a feature, an observation, an evidence, etc. Rarely is only one sensor needed, however. Most applications use several sensors of different kinds, which we refer to as sensor sets. As sensors become more prevalent and ubiquitous, they present their own issues of cost effectiveness and timeliness. It becomes critically

important to select sensor sets that provide the most information at the least cost in a timely and efficient manner. For example, in fault diagnosis problems (medical diagnosis, computer system troubleshooting, etc.) [8], [49], a set of informative tests needs to be selected to provide an optimal tradeoff between the cost of performing the tests and the accuracy of diagnoses. In sensor networks [11], [26], [27], a subset of sensors needs to be selected to achieve a suboptimal tradeoff between the energy consumption of operating the sensors and the information gain. In pattern recognition [21], [33], [42], good features need to be selected to guarantee the performance of the classifiers.

Purposely choosing an optimal subset from multiple sensing sources can save computational time and physical costs, avoid unnecessary or unproductive sensor actions, reduce redundancy, and increase the chance of making correct and timely decisions. Because of these benefits, sensor selection plays a particularly important role for time-critical and resource-limited applications, including computer vision, control systems, sensor networks, diagnosis systems, and many military applications.

Basically, sensor selection problems can be divided into two types. The first type, called a budget-limit case, involves choosing a sensor set with maximum information gain given a budget limit. The other type, called an optimal tradeoff case, involves deciding a sensor set that achieves an optimal tradeoff between the information gain and the cost. Unfortunately, both of them are NP-hard, since the number of sensor subsets grows exponentially with the total number of sensors.

Most work formulates sensor selection as an optimization problem based on information-theory or decision-theoretic criteria. However, solving the optimization problem efficiently is difficult since the search space usually is large. To be practical, some methods [11], [26], [45] select the best sensor myopically or select the first m sensors greedily. However, the selected sensors could lead to poor performance, since the selection methods cannot provide performance guarantees. Recently, some sensor selection algorithms have been proposed to achieve a balance between performance and efficiency. Isler and Bajcsy [18] present an approximation algorithm for sensor selection that aims to minimize the error in estimating the position of a target within a sensor network. They prove that at least one of the sensor subsets whose sizes are less or equal to six can guarantee that the resulting estimation error is within a factor of two of the least possible error under certain assumptions. Thus, the algorithm is to enumerate all l -subsets ($l \leq 6$) of sensors and choose the best one. Zheng *et al.* [49] use a greedy test-selection strategy to find an optimal subset of tests in a fault diagnosis system. Similarly, this paper provides a theoretical

Manuscript received November 27, 2006; revised October 31, 2007. First published April 17, 2009; current version published June 19, 2009. This paper was recommended by Associate Editor E. P. Blasch.

W. Liao is with the R&D, Thomson-Reuters Corporation, Eagan, MN 55123 USA (e-mail: wenhui.liao@thomsonreuters.com).

Q. Ji is with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: jiq@rpi.edu).

W. A. Wallace is with the Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: wallaw@rpi.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSMCA.2009.2014168

justification for the greedy test-selection approach, along with some performance guarantees (the expected number of tests produced by the greedy strategy is within an $O(\log N)$ factor from the optimal solution). Krause and Guestrin [27] present an efficient greedy approach to select the most informative subset of variables in a graphical model for sensor networks. The algorithm provides a constant factor $(1 - 1/e - \epsilon)$ approximation guarantee for any ϵ with high confidence. However, both algorithms of Zheng and Krause assume that the sensors are conditionally independent given the hypotheses (i.e., the event/situation that sensor fusion aims to assess).

We seek to achieve a balance between the performance and the efficiency of the proposed sensor selection algorithms by fully utilizing the properties of submodular functions and the probabilistic relationships among sensors. The probabilistic relationships between the sensors and the hypotheses are modeled by a Bayesian network (BN). In addition, the informativeness of the sensors with respect to the hypotheses is measured by mutual information. We then prove that mutual information is a submodular function under several conditions. Based on the theory of submodular functions, in the budget-limit case, the proposed sensor selection algorithms provide a constant factor of $(1 - 1/e)$ guarantee to the optimal performance. Furthermore, we propose a partitioning procedure by exploiting sensor dependence embedded in the BN model to compute mutual information efficiently as well as to reduce the search space, so that the efficiency of the algorithms is further improved. In the optimal tradeoff case, a submodular–supermodular procedure is embedded within the proposed sensor selection algorithm to decide the optimal or near-optimal sensor set that maximizes the difference between benefit and cost with polynomial-time complexity.

The following sections are organized as follows. Section II gives a brief overview of a related work. Section III presents the BN model for sensor selection and fusion. Section IV introduces the sensor selection criteria. The sensor selection algorithms for the budget-limit case and the optimal-tradeoff case are described in Sections V and VI, respectively. Section VII discusses the experimental results based on synthetic data, and Section VIII provides an illustrative application. This paper ends in Section IX with a summary and some suggestions for future work.

II. RELATED WORK

Sensor selection usually consists of two problems: selection criterion definition and sensor set selection based on the defined sensor selection criterion. In general, sensor selection can be treated as an optimization problem: finding an optimal sensor set that maximizes the objective function defined by the selection criterion. Most existing work can be divided into two groups based on the sensor selection criterion they used: information-theoretic methods and decision-theoretic methods.

A. Sensor Selection Criteria

Information-theoretic methods apply information theory to define the objective function for sensor selection. The common functions include Shannon's entropy, mutual information, en-

tropy difference, and Kullback–Leibler's (KL) cross-entropy. Hintz [17] uses the expected change in Shannon entropy when tracking a single target moving in one dimension with Kalman filters. Zhang *et al.* [48] apply mutual information to select an optimal sensor set based on a dynamic BN. Denzler and Brown [9] use mutual information to determine the optimal action (set of camera parameters, including focal length, pan, and tilt angles) that will maximally decrease the uncertainty of the object-state estimation process. Guo and Nixon [16] use mutual information to select feature subsets for gait recognition. Wang *et al.* [45] propose an entropy-based sensor selection heuristic for target localization. Instead of using mutual information, they use maximal entropy difference as the criterion to choose one sensor at each step until the required uncertainty level of the target state is achieved. The method is computationally simpler than the mutual-information-based approaches, and their experiments demonstrate that the method can sort candidate sensors into exactly the same order as the mutual-information method does in most cases.

An active sensing approach, based on an entropy measure called the Rényi divergence, is proposed by Kreucher *et al.* [28] to schedule sensors for multiple-target tracking applications. At each time step, only one sensor action is chosen to provide measurement and thus update the probability density of the target states. KL's cross-entropy is used for optimal multisensor allocation [35] and sensor management [24]. Ertin *et al.* [11] employ expected posterior entropy to choose the next measurement node (sensor) in a distributed Bayesian sequential estimation framework. They show that minimizing the expected posterior uncertainty is equivalent to maximizing the mutual information between the sensor output and the target state.

Although information-based criteria (entropy, mutual information, etc.) are the most commonly used functions for ranking information sources in terms of uncertainty reduction, these criteria require intensive computations, particularly when computing for multiple sensors. Therefore, the research reported in the aforementioned papers often selects only one sensor at one time with the information-theoretic criteria. We propose an efficient procedure in this paper to efficiently compute mutual information by exploiting the statistical independences among sensors.

For decision-theoretic approaches, the objective function is defined based on decision theory, where the goal is to choose the optimal sensory action by maximizing an overall utility function (e.g., tradeoff between cost and benefit). Wu and Cameron [47] describe a mathematical framework using Bayesian decision theory to select optimal sensing actions for achieving a given goal, for example, for object recognition in robot vision applications where one optimal action is decided at each time step. Rimey and Brown [41], [40] build a task-oriented computer vision system that uses BN and a maximum-expected-utility decision rule to choose a sequence of task-dependent and opportunistic visual operations on the basis of their utilities.

van der Gaag and Wessels [44] build a decision tree for selective gathering of evidence for diagnostic belief networks. Lindner *et al.* [32] estimate the expected utility of each sensor to predict the minimum cost subset of sensors for a mobile robot application. Kristensen [29] develops a Bayesian decision tree

to solve the problem of choosing proper sensing actions from a family of candidates. Each sensing action is evaluated by its expected interest from sample information (EISI). The Bayesian decision rule simply selects the one with the maximum EISI value as the sensing action to be performed one at a time.

The most common decision-theoretic criterion is the value of information (VOI). The VOI of a sensor set is defined as the difference in the maximum expected utilities with and without the information collected from the sensor set. It evaluates the utility of the sensor set by considering both the benefit and cost of using the sensors. The cost of operating the sensor for evidence collection includes the computational cost, physical cost, etc., while the benefits include financial benefits, performance improvements, reduced loss, etc. Oliver and Horvitz [37], [38] apply an expected-value-of-information (EVI)-based policy to selective perception in SEER, a multimodal system of recognizing office activity that relies on a layered hidden Markov model (LHMM) representation. Although the system uses a dynamic programming strategy to compute the EVI of each feature combination based on the LHMM, it needs to enumerate all the feature combinations and then selects the best one. The process is therefore time-consuming.

A special group in the decision-theoretic methods is based on Markov decision process (MDP). Bayer-Zubek [2] formalizes the diagnosis process as an MDP to find an optimal diagnostic policy that achieves optimal tradeoffs between the costs of tests and the costs of misdiagnoses. Cassandra *et al.* [3] model the problem of deciding optimal actions for mobile-robot navigation as a POMDP. Castanon [4] formulates the problem of dynamic scheduling of multimode sensor resources for classifying a large number of stationary objects as a POMDP.

The decision-theoretic methods provide a precise formulation for sensor selection problems. These measures are usually different from the information-theoretic measures such as mutual information since they directly relate sensor/resource allocation to decision making, while information theoretic criterion relates sensor management to situation/event assessment. On the other hand, like the information-theoretic criteria, decision-theoretic criteria also require significant computation for multiple sensors. As a result, like the information-theoretic criteria, most of the aforementioned methods adopt the myopic approach, i.e., choose only one sensing action at each step. For example, MDP-based approaches suffer from combinatorial explosion when solving practical problems of even moderate size. Another problem of the decision-theoretic methods is the subjective definition of utility functions. The utilities are usually problem dependent and may vary as their environment changes. In addition, for some applications, it may not be possible to derive appropriate utilities.

B. Optimization Methods for Sensor Selection

For both the information-theoretic and decision-theoretic criteria, optimization methods are needed to identify the optimal sensor set. Methods for solving the optimization include search methods and mathematical programming (e.g., approximate dynamic programming, integer programming, and greedy selection strategies). One option is to use heuristic searches.

Heuristic searches use some function that estimates the cost from the current state to the goal, presuming that such a function is efficient. Heuristic search techniques incorporate domain knowledge to improve efficiency over blind search. Some heuristic searches can guarantee an optimal solution, but they could be very slow. Thus, another search strategy is to give up completeness and risk losing optimal subsets in exchange for efficiency, such as *sequential forward selection* [46], *floating search selection* [39], and simulated annealing.

However, due to the high computational cost of search algorithms, in most applications involved with sensor selection, the greedy strategy is used. This strategy can be regarded as the simplest form of sequential search, where, at each iteration, the best sensor is incorporated into the candidate sensor set until there is no improvement in the value of the objective function. A more complex sequential search approach, called entropy adaptative aggregation algorithm, is proposed in [12]. It includes an aggregative phase to heuristically choose the initial subset and an adaptative phase to iteratively aggregate and disaggregate the current subset until it converges. Kalandros *et al.* [23] explore the use of randomization and superheuristics search [31] for sensor selections in target-tracking applications. The search begins with a base sensor set and then generates more alternative solutions via a probabilistic assignment rule. The best solution is decided only from the generated solutions, and so, it may not be optimal. Kundakcioglu and Unluyurt [30] integrate concepts from one-step look-ahead heuristic algorithms and basic idea of Huffman coding to construct a minimum-cost AND/OR decision tree bottom-up for sequential fault diagnosis. In [1], Amari and Pham develop a method to provide lower and upper bounds on the optimal number of spares for each subsystem of complex repairable systems. In this way, the search space is reduced dramatically and a near-optimal solution can be found efficiently.

Our work differs from the foregoing work in that it explicitly exploits the theory of submodular functions with respect to the sensor selection criterion, and the probabilistic relationships among sensors to achieve both efficient and accurate sensor selections in two typical scenarios: the budget-limit case and the optimal-tradeoff case. Hence, the proposed algorithms are not only efficient but also they provide performance guarantees.

III. SENSOR SELECTION AND FUSION MODEL

We use dynamic BNs, as shown in Fig. 1, to model the relationships between sensors and the hypotheses the sensors aim to assess. Given the dynamic Bayesian network (DBN), sensor fusion is performed through probabilistic inference. A BN is a directed acyclic graph that represents a joint probability distribution among a set of variables [5], [22]. In a BN, nodes denote variables and the links between nodes denote the conditional dependences among the variables. The dependence for each node is characterized by a conditional probability table. A DBN additionally models the temporal relationships of the variables. Such a model is capable of representing the relationships among different sensors in a coherent and unified hierarchical structure, accounting for sensor uncertainties and dependences, modeling the dynamics in situation development

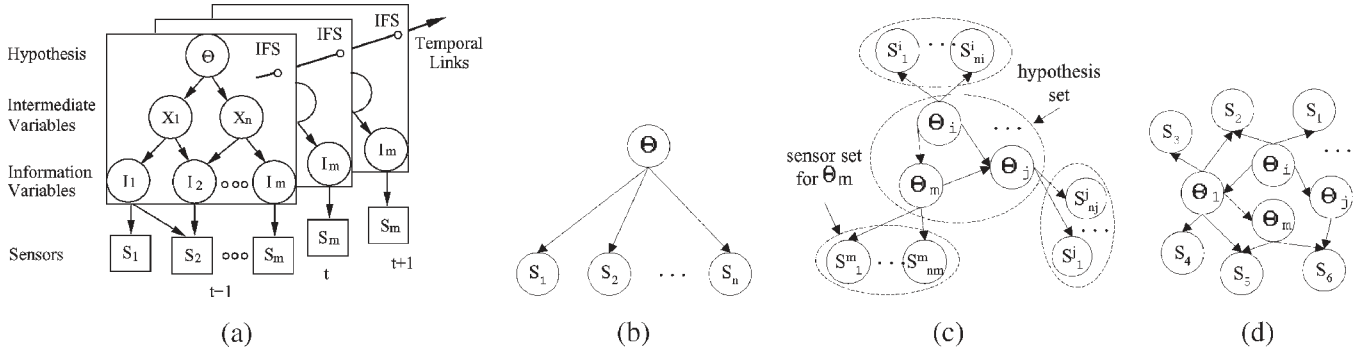


Fig. 1. (a) Dynamic BN for active sensor selection and fusion, where Θ , X , I , and S denote hypothesis variables, intermediate variables, information variables, and sensors, respectively. (b) All the hypotheses shares the same sensors (the naive BN). (c) Each hypothesis has its own individual sensor set. (d) Each hypothesis has individual sensors and/or shares sensors with other hypotheses. The BN in (a) is more appropriate for high-level sensor fusion, while (b), (c), and (d) can be regarded as special cases of the BN in (a) and are more appropriate for low-level sensor fusion.

with temporal links, and providing principled inference and learning techniques to systematically combine the domain information and statistical information extracted from the sensors. These capabilities make DBN a good choice to model sensor selection and fusion.

As shown in Fig. 1(a), the root node Θ of such a network, named as the hypothesis node, represents a mutually exclusive and exhaustive set of possible hypotheses about the events/situations we want to assess. For example, Θ could be the system states in a fault diagnosis system, class labels for a classification problem, enemy intents in a battlefield situation-assessment scenario, etc. Sensors occupy the lowest level nodes without any children. Evidence is gathered through sensors. To model the integrity/reliability with sensory readings, an information node I_i is introduced for each sensor S_i . I_i contains the information that sensor S_i measures. The conditional probabilities between the information node I_i and the corresponding sensor node S_i quantify the reliability of sensor measurements, and the sensor reliability may change over time. The intermediate nodes X 's model the probabilistic relations between the hypothesis and information nodes at different abstraction levels. The DBN in Fig. 1(a) represents a typical structure for active sensor selection and fusion using BN. However, the BN structure could be more flexible. For example, Θ is not necessarily a root node, and there may be multiple Θ 's. In addition, S_i is not necessarily a leaf node either, and the intermediate nodes X 's are not needed for some applications. Fig. 1 (b)–(d) shows the different BN configurations for sensor fusion.

IV. INFORMATION GAIN AND SENSOR COST

The goal of sensor selection is to select a group of sensors that achieves a good balance between the information gain (benefit) and the cost of the sensors. In recent years, a commonly used method to measure information gain is mutual information. With respect to Fig. 1(a), let S be a sensor set, $S = \{S_1, S_2, \dots, S_n\}$; the mutual information of S with respect to Θ , $I(\Theta; S)$, is defined as follows: $I(\Theta; S) = H(\Theta) - H(\Theta|S)$, where H indicates the entropy function. $I(\Theta; S)$ measures the uncertainty reduction of Θ given the sensors in S .

Mutual information may have a very interesting property, called *submodularity*. Before we prove it, we first give the background information about submodular functions.

Let V be a finite set, and let f be a set function: $2^V \mapsto \mathfrak{R}$. A function f is *submodular* if the inequality

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$$

holds for every pair of sets $A, B \subseteq V$ [34]. Equivalently

$$f(A \cup X) - f(A) \geq f(B \cup X) - f(B)$$

for $\forall A \subset B \subset V, X \in V$. It means that the marginal value of X with respect to A is larger than the marginal value of X with respect to a larger set such as B . In other words, for submodular functions, adding X into a smaller set helps more than adding it into a larger set. For example, entropy functions, cut capacity functions, and matroid rank functions are submodular.

The negative of a submodular function is called a *supermodular* function. In other words, a function f is supermodular if the inequality

$$f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$$

holds for every pair of sets $A, B \subseteq V$. In addition, a function that is both submodular and supermodular is called a *modular* function.

In general, mutual information is not a submodular function [27]. However, we show that it is submodular under several conditions.

Proposition 1: Let A be any subset on S and $f(A) = I(\Theta; A)$. If any sensor in S is conditionally independent of each other given Θ , then f is submodular.

Proof: See Appendix.

Although people usually make the assumption of conditional independence in their frameworks such as in [27] and [49], it is still a strong assumption for most applications. Proposition 2 shows that f is still a submodular function under a relaxed assumption.

Proposition 2: Let A, B be any subset on S and $f(A) = I(\Theta; A)$. If $I(\Theta; B \setminus A|A \cap B) \geq I(\Theta; B \setminus A|A)$ or $I(\Theta; A \setminus B|A \cap B) \geq I(\Theta; A \setminus B|B)$, then f is a submodular function.

Proof: See Appendix.

$I(\Theta; B \setminus A|A \cap B) \geq I(\Theta; B \setminus A|A)$ shows that the mutual information between Θ and the sensors in $B \setminus A$ is related to how much information is already known. It is reasonable

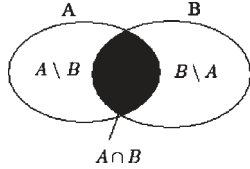


Fig. 2. Graph illustration of Proposition 2: $I(\Theta; B \setminus A | A \cap B) \geq I(\Theta; B \setminus A | A) \Leftrightarrow I(\Theta; A \setminus B | A \cap B) \geq I(\Theta; A \setminus B | B)$. The “\” sign means set subtraction.

to assume that observing the sensors in $B \setminus A$ reduces the uncertainty of Θ less if more sensors have already been observed. Thus, the mutual information between Θ and the sensors in $B \setminus A$ given A is less than that when a smaller set $A \cap B$ is observed. Fig. 2 shows such a concept.

In addition, the following proposition shows that f is a nondecreasing function.

Proposition 3: Let A be any subset on S . If $f(A) = I(\Theta; A)$, then f is nondecreasing and $f(\emptyset) = 0$.

Proof: See Appendix. We will show that the submodularity and nondecreasing of the information gain function help to provide a performance guarantee to our sensor selection algorithms.

The cost function for each sensor or sensor set can be simple or complex. The sensor cost could be computational cost, operational cost, cost of energy consumption, and others. The cost could be constant and the same for different sensors, or the cost could vary with the sensors. How to define the cost is application dependent. In order to simplify the sensor selection problems, most applications assume that each sensor S_i has a constant cost $c(S_i)$, and the cost of a sensor set A_i is $\sum_{S_i \in A_i} c(S_i)$. However, it is often the case that the cost of operating a collection of sensors (jointly) is less than the combined cost of operating each sensor individually. In that case, it is reasonable to model c as a submodular function.

No matter how the benefit and cost are defined, the sensor selection problems can be divided into two categories. One involves finding a best sensor set with maximum information gain when the cost is within a budget limit. Another involves finding a best sensor set that achieves the best tradeoff between the information gain and the cost. Since both of these categories play an important role in the vast majority of applications, we propose efficient sensor selection algorithms for both of them.

V. SENSOR SELECTION WITH A BUDGET LIMIT

In this section, we present the sensor selection algorithms for the budget-limit case. In the next section, we will present the selection algorithms for the optimal-tradeoff case. Let $S = \{S_1, S_2, \dots, S_n\}$ indicate the available sensors, and let $A = \{A_1, \dots, A_m\}$ be a collection of sets over S . The cost of each A_i is defined as $c(A_i) = \sum_{S_i \in A_i} c(S_i)$, and the benefit of each A_i is $f(A_i) = I(\Theta; A_i)$. Given a budget-bound L , the optimal sensor set A^* is

$$A^* = \arg \max_{A_i \in A} \{f(A_i) : c(A_i) \leq L\}. \quad (1)$$

TABLE I
PSEUDOCODE OF ALGORITHM 1: GREEDY-BRUTE FORCE

<p>Algorithm 1: $A^* = \text{SensorSelect1}(BN, S, L, k)$</p> <p>Let $G = \{A_i : A_i = k, c(A_i) \leq L, A_i \in A\}$; $A_1^* \leftarrow \arg \max_{A_i} \{f(A_i) : A_i < k, A_i \in A, c(A_i) \leq L\}$; $A_2^* \leftarrow \emptyset$; for each $A_i \in G$ $G' \leftarrow S \setminus A_i$; while $G' \neq \emptyset$ $S^* \leftarrow \arg \max_{S_i} \left\{ \frac{f(A_i \cup S_i) - f(A_i)}{c(S_i)} : S_i \in G' \right\}$; if $c(A_i) + c(S^*) \leq L$ then $A_i \leftarrow A_i \cup \{S^*\}$; $G' \leftarrow G' \setminus S^*$; if $f(A_i) > f(A_2^*)$ then $A_2^* = A_i$; Return $A^* = \arg \max(f(A_1^*), f(A_2^*))$;</p>

A. Initial Greedy Algorithm

Based on the aforementioned definition, the sensor selection problem can be regarded as a budgeted maximum coverage problem [43]. Although, in general, such a problem is NP-hard, an approximate solution is available, which is near optimal and computable in polynomial time. We modified the algorithm from [27] as shown hereinafter. Similar algorithms can also be found in [25] and [43].

Table I illustrates the pseudocode of Algorithm 1. In the first phase, Algorithm 1 arrives at a solution A_1^* by enumerating all possible l -element ($l < k$) subsets that satisfy the budget constraint. In the second phase, Algorithm 1 starts from all possible k -element subsets for some constant k and then uses a greedy approach to supplement these sets in order to produce a solution A_2^* . Finally, the algorithm outputs A_1^* if $f(A_1^*) > f(A_2^*)$ and A_2^* otherwise. The time complexity of the algorithm is $O(n^{k+1} \gamma \log n)$, where n is the size of the whole sensor set and γ is the time required to compute the function value of f .

Theorem 1: The worst case performance guarantee of Algorithm 1 for solving the problem (1) is equal to $(1 - 1/e)$, if $k \geq 3$ and f is a submodular and nondecreasing function.

Theorem 1 shows that the information gain of any sensor set selected by Algorithm 1 will not be less than $(1 - 1/e)$ of the information gain of the optimal sensor set. The theorem has been proven by the studies in [13], [27], and [43].

Since f is a submodular and nondecreasing function as we proved in the previous section, Algorithm 1 can be used to find a near-optimal solution for problem (1) in polynomial time with a performance guarantee.

B. Algorithm Speedup With Partitioning

In Algorithm 1, for each updated A_i , we need to compute $f(A_i \cup S_i)$ for each possible candidate sensor S_i at each loop. As the size of A_i increases, the computation could be time-consuming. Therefore, it is important to speed up Algorithm 1. We propose a partitioning procedure to compute f efficiently by exploiting the probabilistic dependence among the sensors.

Assume that the target is to compute $f(S) = I(\Theta; S_1, \dots, S_n)$. The partitioning procedure is to divide S into several groups, where the sensors in one group are conditionally independent of the nodes in other groups given Θ . The partitioning procedure consists of three steps.

- 1) Decide whether two sensors S_i and S_j are conditionally independent given Θ by exploring the BN structure based on four rules: i) if there is a path between S_i and S_j without passing Θ , S_i and S_j are dependent; ii) if both S_i and S_j are the ancestors of Θ , S_i and S_j are dependent given Θ ; iii) after removing the links to and from Θ from the original BN, if S_i and S_j have common ancestors, or S_i is S_j 's ancestor, or vice versa, then S_i and S_j are dependent; and iv) in all the other cases, S_i and S_j are conditionally independent given Θ (time complexity for step 1): $O(h)$, where h is the longest path in a BN).
- 2) Build an undirected graph to model the relationships among the sensors. In such a graph, each vertex represents a sensor S_i , and each edge between two vertices indicates that the two corresponding sensors are dependent according to the rules in Step 1).
- 3) Partition the graph into disjoint connected subgraphs. A depth first search algorithm [6] is used to partition the graph into several *connected components* (disjoint connected subgraphs) so that each component is disconnected from other components. The sensors in each connected component are conditionally independent of the sensors in any other connected components. Therefore, each connected component corresponds to one group [time complexity for step 3: $O(|V| + |E|)$, where V is the set of vertices and E is the set of edges in the graph].

Based on the time complexity, we can see that the partitioning procedure is quite efficient. In addition, we only need to perform the partitioning procedure once for the whole sensor set. To divide the sensors in any subset into several groups, we need only look at the subgraph related with the targeted sensors. Thus, the whole procedure is efficient. The partitioning procedure returns several independent groups $A_c^1, A_c^2, \dots, A_c^m$. Then, the following lemmas stand.

Lemma 1: $H(\Theta, S_1, \dots, S_n) = \sum_{i=1}^m H(A_c^i|\Theta) + H(\Theta)$.

Proof: See Appendix.

Lemma 2: The mutual information $I(\Theta; S_1, S_2, \dots, S_n) = H(A_c^1, \dots, A_c^m) - \sum_{i=1}^m H(A_c^i|\Theta)$.

Proof: See Appendix.

Based on Lemma 2, computing $I(\Theta; S_1, \dots, S_n)$ is equal to computing $H(A_c^1, \dots, A_c^m)$ and each $H(A_c^i|\Theta)$, $i = 1, \dots, m$. To compute $H(A_c^1, \dots, A_c^m)$ efficiently, one key issue is to compute $p(A_c^1, \dots, A_c^m)$. We notice that

$$\begin{aligned} p(S_1, \dots, S_m) &= \sum_{\Theta} \{p(A_c^1, \dots, A_c^{m-1}, A_c^m|\Theta) p(\Theta)\} \\ &= \sum_{\Theta} \left\{ \prod_i^{m-1} p(A_c^i|\Theta) \cdot p(A_c^m|\Theta) \cdot p(\Theta) \right\}. \end{aligned}$$

Since Algorithm 1 always starts from a smaller sensor set to a larger set, each $p(A_c^i|\Theta)$, $i = 1, \dots, m-1$, is usually already computed before computing $p(A_c^1, \dots, A_c^{m-1}, A_c^m)$.

TABLE II
PSEUDOCODE TO COMPUTE $I(\Theta; S_1, \dots, S_n)$

Partitioning S into conditionally independent groups $A_c^1, A_c^2, \dots, A_c^m$ by the DFS (depth first search) algorithm; for each A_c^i , $i = 1, \dots, m$ if $H(A_c^i)$ is computed before Reuse the values of $p(A_c^i)$ and $H(A_c^i)$ else compute $H(A_c^i)$ compute $\sum_{\Theta} \{\prod_i^{m-1} p(A_c^i \Theta) \cdot p(A_c^m \Theta) \cdot p(\Theta)\}$ compute $H(A_c^1, \dots, A_c^m)$ compute $\sum_{i=1}^m H(A_c^i \Theta)$ compute $I(\Theta; S_1, \dots, S_n)$ based on lemma 2
--

Then, $p(A_c^m|\Theta)$ is the only new factor that must be computed. This factor can be easily computed with a BN inference algorithm. Thus, $p(A_c^1, \dots, A_c^{m-1})$ can be computed much more easily. Similarly, for the second term in Lemma 2, $\sum_{i=1}^m H(A_c^i|\Theta)$, each $H(A_c^i|\Theta)$, $i = 1, \dots, m-1$, is usually already computed, so $H(A_c^m|\Theta)$ is the only term that needs to be computed. Overall, the partitioning procedure allows computation sharing between different sensor sets. In particular, the f function of each large sensor set becomes easy to compute because of sharing computations with its subsets. Therefore, the partitioning procedure is able to speed up Algorithm 1 significantly as shown in the experiments. The pseudocode to compute mutual information $I(\Theta; S_1, \dots, S_n)$ is shown in Table II.

C. New Algorithm With Partitioning

The partitioning procedure helps only with computing mutual information and does not affect the selection procedure in Algorithm 1. However, in order to further speed up Algorithm 1, we can also apply the partitioning procedure to sensor selection and thus get a new selection algorithm. This algorithm exploits both the submodularity property and the sensor dependence embedded in the BN model through the partitioning procedure. The pseudocode is shown in Table III.

Algorithm 2 consists of three phases. In the first phase, it uses the partitioning procedure to divide all the sensors into several groups, $A_c^1, A_c^2, \dots, A_c^m$. Then, in the second phase, for each group A_c^i , a subfunction, Algorithm 1 or a brute-force algorithm, is called to select a local optimal sensor set subject to the local budget L'_i . If the size of the group is small ($\leq l$), the brute-force algorithm is used to find the local optimal set. Otherwise, Algorithm 1 is applied to decide the local optimal set. In the third phase, a new sensor set A^{*l} is constructed by combining all the local optimal sets. For this set, Algorithm 1 or the brute-force method is applied again to decide the global optimal sensor set.

There are two dynamic parameters, l and L'_i , in the algorithm. l decides which algorithm is applied to each group to obtain the local optimal set: the brute-force method or Algorithm 1. If l is too large, the brute-force algorithm will slow down the speed. Empirically, l is set as 6. Another parameter L'_i decides

TABLE III
PSEUDOCODE OF ALGORITHM 2: OPTIMAL
SELECTION WITH PARTITIONING

<p>Algorithm 2: $A^* = \text{SensorSelect2}(\text{BN}, \text{S}, \text{L})$</p> <p>Partitioning S into conditionally independent groups $A_c^1, A_c^2, \dots, A_c^m$ by the DFS (depth first search) algorithm;</p> <p>for each $A_c^i, i = 1, \dots, m$</p> $L_i^i = L(c_1 \frac{n \sum_{S_i \in A_c^i} f(S_i)/c(S_i)}{ A_c^i \sum_{S_i \in S} f(S_i)/c(S_i)} + c_2 \frac{ A_c^i }{n})$ <p>if $A_c^i \leq l$</p> <p style="padding-left: 2em;">use brute-force method to select the best sensor</p> <p style="padding-left: 2em;">set A_i^{*i} from A_c^i under the cost constraint of L_i^i;</p> <p>else</p> <p style="padding-left: 2em;">$A_i^{*i} = \text{SensorSelect1}(\text{BN}, A_c^i, L_i^i, 3)$</p> <p>$A^{*i} = \{A_1^{*i}, \dots, A_m^{*i}\}$;</p> <p>if $A^{*i} \leq l$</p> <p style="padding-left: 2em;">A^* is decided from A^{*i} with brute-force method;</p> <p>else</p> <p style="padding-left: 2em;">$A^* = \text{SensorSelect1}(\text{BN}, A^{*i}, L, 3)$;</p> <p>Return A^*;</p>

the budget for each group. It is dynamically decided by two factors: 1) $(1/|A_c^i|) \sum_{S_i \in A_c^i} f(S_i)/c(S_i)$, the average ratio of $f(S_i)/c(S_i)$ in each group A_c^i , and 2) $|A_c^i|$, the size of A_c^i .

Obviously, the time complexity of Algorithm 2 depends on the two subfunctions: Algorithm 1 and the brute-force method. For the brute-force method, since the size of the input group is always less than l ($l \leq 6$), it is time efficient. Since the input is a subset when calling Algorithm 1, it costs much less time than when the input is the whole sensor set. Therefore, Algorithm 2 is more efficient than Algorithm 1 in general. If all the sensors are conditionally independent of one another, Algorithm 2 degenerates to Algorithm 1.

In addition, the performance of Algorithm 2 can be justified by two factors: 1) algorithm 2 performs sensor selection on each sensor subset, instead of starting from the whole sensor set; thus, it is more efficient; and 2) the sensor subsets are conditionally independent from each other through partitioning; thus, the sensors selected from each subset are more likely to be included in the optimal sensor set, compared to dependent sensor subsets. Therefore, Algorithm 2 significantly improves the speed of Algorithm 1, yet with comparable accuracy.

VI. SENSOR SELECTION WITH OPTIMAL TRADEOFF

The algorithms in Section V maximize the benefit of a set of sensors as long as the cost is within a budget. A further goal is to find a sensor set with the best tradeoff between the cost of the sensors and the potential gain (benefit) obtained by using the sensors. In certain applications, it is more reasonable to achieve the optimal sensor set A^* such that

$$A^* = \arg \max_{A_i} \{f(A_i) - c(A_i)\}. \quad (2)$$

Taking sensor networks for example, c could be sensor power usage. It is desirable to have sensors that maximize the gain while consuming as little power as possible in sensor networks. This is an NP-hard problem too. Based on the previous analysis, f is a nondecreasing submodular function. If c is a modular function as assumed by most researchers, $f(A_i) - c(A_i)$ would still be a submodular function but not necessarily a nondecreasing one. Therefore, problem (2) becomes a submodular function maximization problem, which can be solved with global solutions by some well-known polynomial algorithms such as the combinatorial strongly polynomial algorithm Iwata, Fleischer, and Fujishige (IFF) developed by Iwata *et al.* [20], a faster scaling algorithm in [19], etc. However, in some cases, it is not reasonable to define c as a modular function. For example, usually, the cost of operating a collection of sensors (jointly) is less than the combined cost of operating each sensor individually. Thus, assuming that the cost of a sensor set is the sum of the cost of each individual sensor may not be appropriate. In that case, it is more practical to model c as a submodular function.

Unfortunately, if c is also a submodular function, the difference between two submodular functions like f and c is not necessarily a submodular function. However, if we could transfer $f(A_i) - c(A_i)$ into a submodular function, the optimal sensor set A^* can be obtained efficiently with one of these well-known algorithms. One solution is to seek a modular function h that closely approximates the cost function c . Since h is a modular function, $f(A_i) - h(A_i)$ will still be a submodular function. Proposition 4 shows how to find such a function h , which is referred from the studies in [10] and [36].

Proposition 4: Let π be a random permutation of the sensor set $S = \{S_1, \dots, S_n\}$. Let $W_i = \{\pi(1), \pi(2), \dots, \pi(i)\}$. A function $h : S \rightarrow \mathfrak{R}$ is defined as follows:

$$h(\pi(i)) = \begin{cases} c(W_1), & \text{if } i = 1 \\ c(W_i) - c(W_{i-1}), & \text{otherwise.} \end{cases}$$

In addition

$$h(A_i) = \sum_{S_i \in A_i} h(S_i) \quad \forall A_i \subseteq S.$$

Then, the following are defined.

- 1) $h(A_i) \leq c(A_i)$ for $\forall A_i \subseteq S$.
- 2) $h(W_m) = c(W_m)$, for $1 \leq m \leq n$.

Proof: See [15].

With this procedure, the generated function h is modular and bounded above by c . Furthermore, this modular approximation is the tightest possible approximation to c in the following sense [36].

Proposition 5: Every h obtained in Proposition 4 is a vertex in the extended base polymatroid of c , and every vertex in the extended base polymatroid of c can be obtained by picking an appropriate permutation.

Proof: See [15].

Now, we can use Algorithm 3, as depicted in Table IV, to decide the optimal or near-optimal solution for the problem (2) by

TABLE IV
PSEUDOCODE OF ALGORITHM 3: SELECTION
WITH MODULAR APPROXIMATION

Algorithm3: SensorSelect3(BN, S, δ)
$max \leftarrow -\infty;$
$mark \leftarrow true;$
$A^* \leftarrow \emptyset;$
while $mark = true$ do
if A^* is empty
$\pi \leftarrow$ a random permutation of S ;
else
$\pi \leftarrow$ random permutation starting with A^* ;
define h with the method in Proposition 4 given π ;
$A^* \leftarrow \arg \max_{A_i \subseteq S} (f - h)(A_i)$
if $(f - h)(A^*) - max > \delta$
$mark = true;$
$max = (f - h_n)(A^*);$
else
$mark = false;$
Return A^* ;

using Proposition 4 and the submodular function maximization algorithm.

As shown in the pseudocode, it repeats generating new permutation of the sensor set S based on the selected sensor set A^* at the previous step until no obvious improvement is achieved, which is controlled by the parameter δ . Corresponding to each new permutation, a modular function h is generated to approximate c , and $A^* \leftarrow \arg \max_{A_i \subseteq S} (f - h)(A_i)$ can be decided with the IFF algorithm by Iwata *et al.* [20]. The time complexity of IFF algorithm is $O(n^\gamma \log(n))$, where γ is the time required for computing the function value of $(f - h)$.

If the cost function c is a modular function, no modular approximation is necessary, and Algorithm 3 actually returns the global optimal solution. When the cost function c is a submodular function, because of introducing h to approximate c , local optimum can be returned. However, since h is very close to c (as shown in Proposition 5) and we also use δ to control the precision effects, the returned solution is usually optimal or near optimal (as will be shown in the experiments).

VII. EXPERIMENTS

In the experiments, we analyze the performance of the proposed sensor selection algorithms in terms of both accuracy and speed. To demonstrate the robustness of the algorithms to different BN structures and parameters, we generate BNs with random structures and parameters, whose maximal number of nodes is 50. In each BN, 12 nodes are randomly selected as the sensor nodes and 1 node is selected as the hypothesis node. For the budget-limit case, the cost of each sensor is randomly set at each testing case and the budget-limit L is set as a constant for all the cases. For the optimal-tradeoff case, the sensor cost

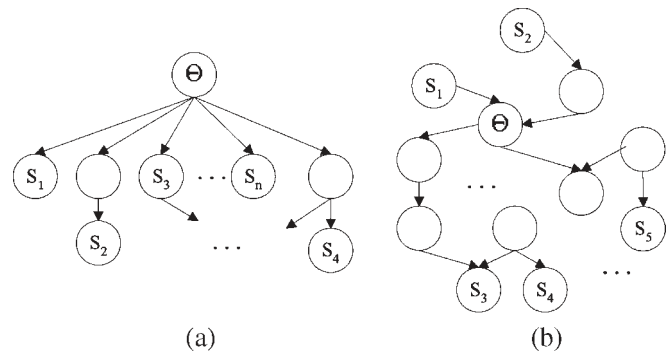


Fig. 3. (a) Example BN where all the sensors are conditionally independent of one another given Θ . (b) Example BN where the sensors can be dependent of one another given Θ .

TABLE V
PERFORMANCE OF THE SENSOR SELECTION ALGORITHMS WHEN
SENSORS ARE CONDITIONALLY INDEPENDENT IN 500 TESTING CASES

	Alg. 1a	Alg. 1b	Greedy
Error Ratio	0.05	0.05	0.21
Mutual Information Ratio	0.98	0.98	0.91
Running Time Ratio	0.35	0.06	0.03

TABLE VI
PERFORMANCE OF THE SENSOR SELECTION ALGORITHMS WHEN
SENSORS ARE DEPENDENT IN 500 TESTING CASES

	Alg. 1a	Alg. 1b	Alg. 2	Greedy
Error Ratio	0.06	0.06	0.10	0.22
Mutual Information Ratio	0.99	0.99	0.97	0.91
Running Time Ratio	0.35	0.10	0.06	0.03

is defined as a submodular function. The ground truth of the optimal sensor subset is obtained by a brute-force approach.

A. Sensor Selection With a Budget Limit

To demonstrate the performance of the proposed algorithms, we compare Algorithm 1, Algorithm 2, a greedy approach, and the brute-force method. For Algorithm 1, k is set as 3. The greedy approach is similar to Algorithm 1 with $k = 1$.

In order to demonstrate whether the performances of the algorithms are affected by the relationships among the sensors, we designed two sets of testing cases. The first set consists of 500 BNs where all the sensors are conditionally independent of one another; and the second set consists of 500 BNs where the sensors can be dependent of one another. Fig. 3 shows two example BNs.

Tables V and VI demonstrate the experimental results, where Alg. 1a represents Algorithm 1 without using the partitioning procedure to compute mutual information, while Alg. 1b represents Algorithm 1 with the partitioning procedure. Table V does not list the performance of Alg. 2 because it has the same performance as Alg.1 when all the sensors are conditionally independent. The *error ratio* is the ratio between the number of misselection cases and the overall number of cases, where a misselection case is defined as the case that the selected sensor

TABLE VII
PERFORMANCE OF ALGORITHM 3 VERSUS GREEDY
APPROACHES IN 500 TESTING CASES

	Error Ratio	DIC rate
Algorithm 3	0	0.99
Greedy approach (k=3)	0.25	0.94
Greedy approach (k=1)	0.42	0.82

set has more than one sensor different from the optimal selection. The *mutual information ratio* is the mutual information between the selected sensors and Θ , divided by the mutual information between the optimal sensor set and Θ . The *running time ratio* is the ratio between the computational time of the algorithm in each column and that of the brute-force method. Both the mutual information ratio and the running time ratio are averaged over the 500 testing cases.

As shown in the tables, the performance of each algorithm is rarely affected by the relationships among the sensors. Compared to the greedy approach, Alg. 1a, Alg. 1b, and Alg. 2 have much lower error ratio, which means that they are able to return the near-optimal solutions for most testing cases. In addition, Alg. 2 is faster than Alg. 1a and Alg. 1b because the partitioning procedure is applied to both mutual information computation and selection procedure. Alg. 1b is faster than Alg. 1a because of the savings in mutual information computation. Greedy approach, on the other hand, is faster than both algorithms 1 and 2 for both types of networks, but with much worse error ratios.

B. Sensor Selection With Optimal Tradeoff

To test Algorithm 3, 500 BNs are randomly generated. However, the cost of sensors is defined as a submodular function. We compare Algorithm 3 to a greedy method, which is very similar to Algorithm 1 except that the budget limit is removed. The overall performance is shown in Table VII. The *DIC rate* is the ratio between the DIC (difference of mutual information and cost) of the selected sensor set and that of the optimal sensor set. Both the DIC and error ratio are averaged over the 500 testing cases.

Table VII shows that the error ratio of Algorithm 3 is zero. It demonstrates that Algorithm 3 always returns an optimal or near-optimal solution. For the greedy approach, although it has good performance for the budget-limit case since the objective function is submodular and nondecreasing, it performs worse in the optimal tradeoff case. Since $(f - c)$ is neither a submodular function nor nondecreasing, there is no performance guarantee for the greedy approach even with $k \geq 3$ in the optimal tradeoff case. Fig. 4 shows the performance of Algorithm 3 versus greedy approaches in 50 cases. In the majority of the cases, the DIC is 1, which means that the proposed algorithm returns the optimal sensor sets, while the greedy approaches rarely return the optimal sensor sets.

When n is small, Algorithm 3 is not much faster than the brute-force method, although the time complexity of the former is polynomial and the latter is exponential. However, when n is large (e.g., $n \geq 25$), it is much faster than the brute-force

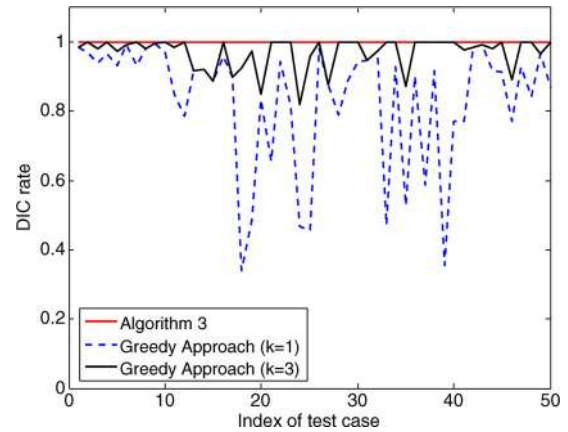


Fig. 4. Performance of Algorithm 3 versus greedy approaches. The x -coordinate is the test case index, and the y -coordinate is the DIC rate.

method, in addition to its capability of providing optimal or near-optimal solutions.

VIII. ILLUSTRATIVE APPLICATION

We apply the proposed sensor selection algorithms to an application of multistage battlefield situation assessment. The scenario here is inspired from the study in [7]. In this application, dynamic BN is used to model the sensors and hypothesis. Although we focus on static BNs in the previous sections, our algorithms can also apply to dynamic BNs one frame at a time, as shown in this application.

The scenario develops during a period of growing hostility between the Blue force and the Red force who poses a threat. The goal of this scenario is for the Blue force to selectively use its surveillance assets to quickly and efficiently infer the intent of the Red force. More detail is given in [7]. The Blue force surveillance facilities include a number of offshore sensors, unmanned aerial vehicles, surveillance helicopters (RAH66 Comanche), etc. The Blue forces on duty in the restricted zone consist of the following: 1) a Fremantle Class Patrol Boat (FCPB); 2) a Maritime Patrol Aircraft (MPA); 3) a Night Hawk Helicopter; and 4) one F111 (Maritime Strike Aircraft). The Red forces include the following: 1) a major fleet unit; a Guided Missile Frigate (FFG); 2) one FCPB; and 3) a communication ship. In addition, the Red force has two surface units armed with an M386 Rocket Launcher and a 110 SF Rocket Launcher that are ready to move to the locations where the Blue force is within their fire range.

A dynamic BN, as shown in Fig. 5, is constructed to assess the situations for the scenario above. A set of hypotheses representing possible Red force intents includes the following: 1) Passive—monitor the Blue forces in the restricted zone and assume that the Blue forces will not interfere with the fuel supply; 2) Defensive—conduct active reconnaissance and maintain a defensive presence to guard the supply routes against the Blue force interference; and 3) Offensive—mount a naval attack or infantry artillery engagement (surface-to-air or surface-to-surface attack) on the Blue forces with the intent of destroying the Blue forces as well as their offshore surveillance facilities.

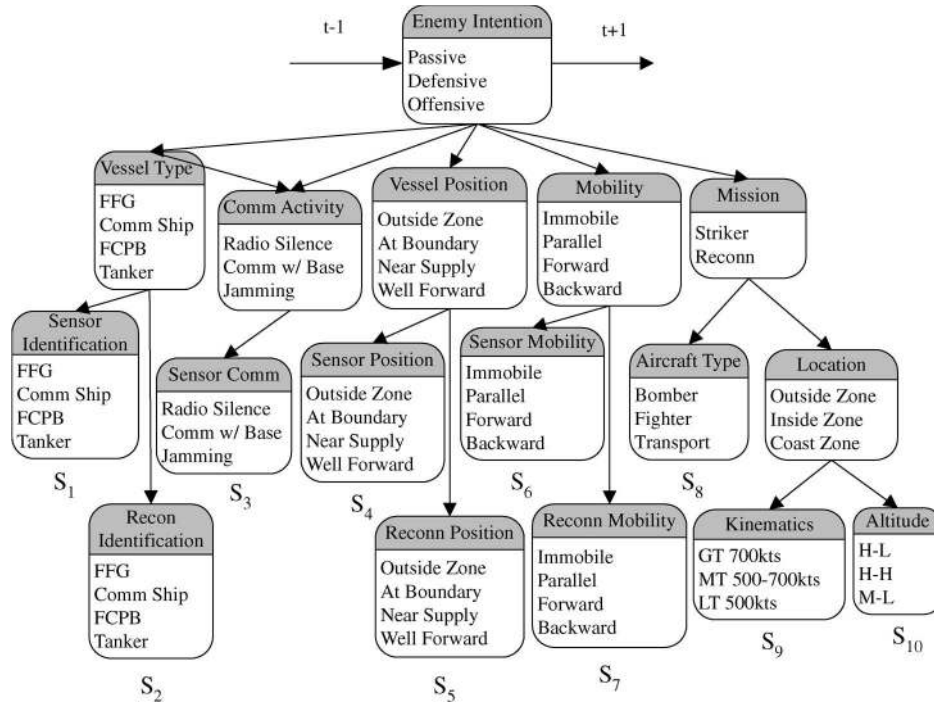


Fig. 5. BN model for the battlefield scenario. $S_1 - S_{10}$ are information sources.

We assume that there are external modules that receive sensor data and make the data available as input evidence to the network. The model in Fig. 5 shows that there are ten sensors available to supply information. The conditional probabilities and sensor costs are given subjectively. A Blue force commander needs to select appropriate sensors over time in order to assess the hypothesis of the Red force intent (Passive, Defensive, or Offensive in a timely and efficient manner).

In order to further demonstrate how the sensor selection algorithms help predict enemy intents, a simulation system is developed to generate synthetic data. The simulator consists of two independent but related models: a *source model* simulating the intents of the Red force to produce evidence reflecting them, and a *working model* simulating the Blue force that estimates enemy (Red force) intents and determines appropriate sensory actions by selectively collecting the evidence. We assume that the enemy intents are passive in the beginning, gradually change to offensive, and finally become defensive.

Fig. 6 shows the estimation results when different sensor selection algorithms are used. The dotted line represents the ground-truth enemy intent $P(\text{Offensive})$, and the other three curves represent the inferred enemy intent by collecting evidence from the selected sensors with the corresponding three sensor selection algorithms: Alg. 1b, Alg. 2, and the greedy approach. As shown in the figure, the enemy intent estimated by Alg. 1b and Alg. 2 is quite close to the true enemy intent after 5 time steps, while the enemy intent estimated by the greedy algorithm is not close to the truth until after more than 20 time steps. After about 30 time steps, as $P(\text{offensive})$ decreases, Alg. 1b and Alg. 2 are able to follow enemy intent more closely than the greedy algorithm as well. In other words, Alg. 1b and Alg. 2 are able to select optimal sensors in time and thus track the enemy intent better. In addition, Alg. 2 is

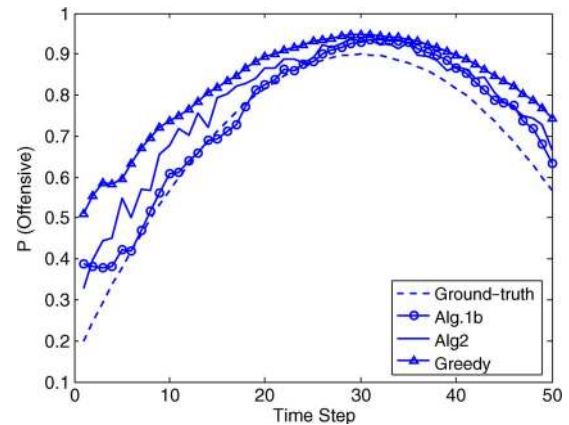


Fig. 6. Sensor selection results of the military example.

TABLE VIII
CASE STUDY OF SENSOR SELECTION WITH ALG. 1b

Assessment Stage	Probability of Hypothesis	Sensors Selected
1	$P(\text{Pas})=0.33$, $P(\text{Def})=0.33$, $P(\text{Off})=0.33$	S_9 , S_{11}
8	$P(\text{Pas})=0.17$, $P(\text{Def})=0.32$, $P(\text{Off})=0.51$	S_9 , S_{11}
19	$P(\text{Pas})=0.01$, $P(\text{Def})=0.18$, $P(\text{Off})=0.81$	S_9 , S_{14} , S_{16}
30	$P(\text{Pas})=0.03$, $P(\text{Def})=0.05$, $P(\text{Off})=0.92$	S_9 , S_{13} , S_{17}
45	$P(\text{Pas})=0.01$, $P(\text{Def})=0.24$, $P(\text{Off})=0.75$	S_9 , S_{14} , S_{16}

Note: Pas, Def, and Off represent Passive, Defensive, and Offensive, respectively.

40% faster than Alg. 1b. Table VIII shows the specific sensors used in certain time slices.

IX. CONCLUSION

In this paper, we propose several algorithms to perform efficient and accurate sensor selection in two typical

scenarios: the budget-limit case in which the best sensor set is the one with maximum information gain under a budget limit, and the optimal-tradeoff case in which the best sensor set is the one that achieves the optimal tradeoff between the information gain and the cost. Although finding an optimal solution is NP-hard for both of them, we introduce efficient and near-optimal solutions by fully utilizing the properties of the sensor selection criterion and the probabilistic dependences among sensors.

Specifically, for the budget-limit case, to ensure performance of the proposed algorithms, we first prove that mutual information is a submodular function under a relaxed condition. Based on this property, we introduce an efficient greedy approach with a constant factor of $(1 - 1/e)$ performance guarantee to the optimal performance. Furthermore, to improve the efficiency of the algorithms, we propose a partitioning procedure for both efficient sensor selection and efficient mutual information computation. For the optimal tradeoff case, if the cost function is a modular function, the proposed algorithm can provide the global optimal solution in polynomial time; if the cost function is a submodular function, a submodular–supermodular procedure is embedded with the proposed sensor selection algorithm to choose the optimal or near-optimal sensor set in polynomial time. The experimental results with both synthetic and real data demonstrate the performance and efficiency of our algorithms.

This paper focuses only on sensor selection. Our future goal is to model sensor selection, sensor fusion, and decision making in a unified framework. More issues will be addressed, e.g., how to fuse the information collected from the sensors efficiently, how to decide the optimal action based on the fused results, and how to learn the parameters of the BN framework. In addition, we will apply the framework as well as the algorithms to more real-world applications.

APPENDIX

Proof of Proposition 1:

$$\begin{aligned} f(A) &= H(\Theta) - H(\Theta|A) \\ &= H(\Theta) - [H(A, \Theta) - H(A)] \\ &= H(\Theta) - (H(\Theta) + H(A|\Theta)) + H(A) \\ &= H(A) - H(A|\Theta). \end{aligned}$$

Thus, $\forall A, B \subseteq S$

$$\begin{aligned} f(A) + f(B) &= H(A) + H(B) - H(A|\Theta) \\ &\quad - H(B|\Theta) + H(A \cap B) + f(A \cup B) \\ &= H(A \cap B) + H(A \cup B) \\ &\quad - H(A \cap B|\Theta) - H(A \cup B|\Theta). \end{aligned}$$

From the fact that Entropy is a submodular function [14], thus $H(A) + H(B) \geq H(A \cap B) + H(A \cup B)$. In addition,

since the sensors are conditionally independent given Θ

$$\begin{aligned} H(A|\Theta) + H(B|\Theta) &= \sum_{S_i \in A} H(S_i|\Theta) + \sum_{S_i \in B} H(S_i|\Theta) \\ &= \sum_{S_i \in A \cap B} H(S_i|\Theta) + \sum_{S_i \in A \cup B} H(S_i|\Theta) \\ &= H(A \cap B|\Theta) + H(A \cup B|\Theta). \end{aligned}$$

Therefore, $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$; in other words, f is a submodular function.

Proof of Proposition 2: Based on the chain rule of mutual information, we find the following:

$$\begin{aligned} I(\Theta; B \setminus A|A \cap B) &= H(\Theta|A \cap B) \\ &\quad - H(\Theta|(B \setminus A) \cup (A \cap B)) \\ &= H(\Theta|A \cap B) - H(\Theta|B) \\ I(\Theta; B \setminus A|A) &= H(\Theta|A) - H(\Theta|(B \setminus A) \cup (A)) \\ &= H(\Theta|A) - H(\Theta|(A \cup B)). \end{aligned}$$

Thus

$$\begin{aligned} I(\Theta; B \setminus A|A \cap B) &\geq I(\Theta; B \setminus A|A) \\ \Rightarrow H(\Theta|A \cap B) - H(\Theta|B) &\geq H(\Theta|A) - H(\Theta|A \cup B) \\ \Rightarrow H(\Theta) - H(\Theta|A) + H(\Theta) - H(\Theta|B) \\ &\geq H(\Theta) - H(\Theta|A \cap B) + H(\Theta) - H(\Theta|A \cup B) \\ \Rightarrow f(A) + f(B) &\geq f(A \cap B) + f(A \cup B). \end{aligned}$$

If $I(\Theta; A \setminus B|A \cap B) \geq I(\Theta; A \setminus B|B)$ is true, the proof is similar. We thus skip the details. In fact, $I(\Theta; B \setminus A|A \cap B) \geq I(\Theta; B \setminus A|A)$ is equivalent to $I(\Theta; A \setminus B|A \cap B) \geq I(\Theta; A \setminus B|B)$. We skip the proof since it is very straightforward.

Proof of Proposition 3: Let X be any sensor belonging to S , but not in A

$$\begin{aligned} f(A \cup X) - f(A) &= H(\Theta) - H(\Theta|A \cup X) \\ &\quad - (H(\Theta) - H(\Theta|A)) \\ &= H(\Theta, A) - H(A) + H(A, X) \\ &\quad - H(\Theta, A, X). \end{aligned}$$

Since entropy is a submodular function [14] $\Rightarrow H(\Theta, A) + H(A, X) \geq H(A) + H(\Theta, A, X)$ [because $(\Theta, A) \cap (A, X) = A$ and $(\Theta, A) \cup (A, X) = (\Theta, A, X)$].

Thus, $f(A \cup X) - f(A) \geq 0$; in other words, f is nondecreasing. It is also obvious that $f(\emptyset) = 0$.

Proof of Lemma 1:

$$\begin{aligned}
H(\Theta, S_1, S_2, \dots, S_n) &= H(\Theta, A_c^1, A_c^2, \dots, A_c^m) \\
&= - \sum_{\Theta, A_c^1, \dots, A_c^m} \{ p(\Theta, A_c^1, \dots, A_c^m) \\
&\quad \times \log p(\Theta, A_c^1, \dots, A_c^m) \} \\
&= - \sum_{\Theta, A_c^1, \dots, A_c^m} \left\{ p(\Theta, A_c^1, \dots, A_c^m) \right. \\
&\quad \left. \times \log \left[p(\Theta) \prod_{i=1}^m p(A_c^i | \Theta) \right] \right\} \\
&= - \sum_{\Theta, A_c^1, \dots, A_c^m} \left\{ p(\Theta, A_c^1, \dots, A_c^m) \right. \\
&\quad \left. \times \left[\sum_{i=1}^m \log p(A_c^i | \Theta) + \log p(\Theta) \right] \right\} \\
&= - \sum_{i=1}^m \sum_{\Theta, A_c^1, \dots, A_c^m} \{ p(\Theta, A_c^1, \dots, A_c^m) \cdot \log p(A_c^i | \Theta) \} \\
&\quad - \sum_{\Theta} p(\Theta) \log p(\Theta) \\
&= \sum_{i=1}^m H(A_c^i | \Theta) + H(\Theta).
\end{aligned}$$

Proof of Lemma 2:

$$\begin{aligned}
I(\Theta; S_1, S_2, \dots, S_m) &= H(\Theta) - H(\Theta | A_c^1, \dots, A_c^m) \\
&= H(\Theta) - [H(\Theta, A_c^1, \dots, A_c^m) - H(A_c^1, \dots, A_c^m)] \\
&= H(A_c^1, \dots, A_c^m) - \sum_{i=1}^m H(A_c^i | \Theta) \text{ (from Lemma 1)}.
\end{aligned}$$

REFERENCES

- [1] S. V. Amari and H. Pham, "A novel approach for optimal cost-effective design of complex repairable systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 3, pp. 406–415, May 2007.
- [2] V. Bayer-Zubek, "Learning diagnostic policies from examples by systematic search," in *Proc. UAI*, 2004, pp. 27–34.
- [3] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, "Acting under uncertainty: Discrete Bayesian models for mobile robot navigation," in *Proc. IEEE/RSJ Int. Conf. IROS*, 1996, pp. 963–972.
- [4] D. A. Castanon, "Approximate dynamic programming for sensor management," in *Proc. 36th IEEE Conf. Decision Control*, 1997, vol. 2, pp. 1202–1207.
- [5] E. Charniak, "Bayesian networks without tears," *AI Mag.*, vol. 12, no. 4, pp. 50–63, 1991.
- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2002.
- [7] B. Das, "Representing uncertainties using Bayesian networks," DSTO Electron. Surveillance Res. Lab., Salisbury South, Australia, Tech. Rep. DSTO-TR-0918, 1999.
- [8] R. Debouk, S. Lafortune, and D. Teneketzis, "On an optimization problem in sensor selection for failure diagnosis," in *Proc. 38th IEEE Conf. Decision Control*, 1999, vol. 5, pp. 4990–4995.
- [9] J. Denzler and C. M. Brown, "Information theoretic sensor data selection for active object recognition and state estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 145–157, Feb. 2002.
- [10] J. Edmonds, "Submodular functions, matroids and certain polyhedra," in *Proc. Calgary Int. Conf. Combinatorial Struct. Appl.*, 1970, pp. 69–87.
- [11] E. Ertin, J. Fisher, and L. Potter, "Maximum mutual information principle for dynamic sensor query problems," in *Proc. 3rd Int. Symp. IPSN*, 2004, pp. 405–416.
- [12] B. Fassinut-Mombot and J. Choquel, "A new probabilistic and entropy fusion approach for management of information sources," *Inf. Fusion*, vol. 5, no. 1, pp. 35–47, Mar. 2004.
- [13] U. Feige, "A threshold of $\ln n$ for approximating set cover," *J. ACM*, vol. 45, no. 4, pp. 634–652, Jul. 1998.
- [14] S. Fujishige, "Polymatroidal dependence structure of a set of random variables," *Inf. Control*, vol. 39, no. 1, pp. 55–72, 1978.
- [15] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, Jun. 1981.
- [16] B. Guo and M. S. Nixon, "Gait feature subset selection by mutual information," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 1, pp. 36–46, Jan. 2009.
- [17] K. J. Hintz, "A measure of the information gain attributable to cueing," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 2, pp. 434–442, Mar./Apr. 1991.
- [18] V. Isler and R. Bajcsy, "The sensor selection problem for bounded uncertainty sensing models," in *Proc. 4th Int. Symp. IPSN*, 2005, pp. 151–158.
- [19] S. Iwata, "A faster scaling algorithm for minimizing submodular functions," *SIAM J. Comput.*, vol. 32, no. 4, pp. 833–840, 2003.
- [20] S. Iwata, L. Fleischer, and S. Fujishige, "A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions," *J. ACM*, vol. 48, no. 4, pp. 761–777, 2001.
- [21] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 153–158, Feb. 1997.
- [22] F. V. Jensen, *Bayesian Networks and Decision Graphs*. New York: Springer-Verlag, 2001.
- [23] M. Kalandros, L. Y. Pao, and Y.-C. Ho, "Randomization and super-heuristics in choosing sensor sets for target tracking applications," in *Proc. 38th IEEE Conf. Decision Control*, 1999, vol. 2, pp. 1803–1808.
- [24] K. Kastella, "Discrimination gain to optimize classification," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 27, no. 1, pp. 112–116, Jan. 1997.
- [25] S. Khuller, A. Moss, and J. Naor, "The budgeted maximum coverage problem," *Inf. Process. Lett.*, vol. 70, no. 1, pp. 39–45, Apr. 1999.
- [26] F. Kobayashi, T. Fukui, F. Arai, T. Fukuda, F. Kojima, M. Onoda, and Y. Hotta, "Sensor selected fusion with sensor selection based gating neural network," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2002, pp. 1482–1487.
- [27] A. Krause and C. Guestrin, "Near-optimal nonmyopic value of information in graphical models," in *Proc. 21st Conf. UAI*, 2005, pp. 324–333.
- [28] C. Kreucher, K. Kastella, and A. O. Hero, III, "Sensor management using an active sensing approach," *Signal Process.*, vol. 85, no. 3, pp. 607–624, Mar. 2005.
- [29] S. Kristensen, "Sensor planning with Bayesian decision theory," *Robot. Auton. Syst.*, vol. 19, no. 3, pp. 273–286, Mar. 1997.
- [30] O. E. Kundakcioglu and T. Unluyurt, "Bottom-up construction of minimum-cost and/or trees for sequential fault diagnosis," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 5, pp. 621–629, Sep. 2007.
- [31] T. W. E. Lau and Y. Ho, "Super-heuristics and their applications to combinatorial problems," *Asian J. Control*, vol. 1, no. 1, pp. 1–13, 1999.
- [32] J. Lindner, R. R. Murphy, and E. Nitz, "Learning the expected utility of sensors and algorithms," in *Proc. IEEE Int. Conf. Multisensor Fus. Integration Intell. Syst.*, 1994, pp. 583–590.
- [33] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, Apr. 2005.
- [34] L. Lovasz, "Submodular functions and convexity," in *Mathematical Programming—The State of the Art*, A. Bachem, M. Grotchel, and B. Korte, Eds. New York: Springer-Verlag, 1983, pp. 235–257.
- [35] R. Mahler, "A unified foundation for data fusion," in *Proc. SPIE—Select. Papers Sensor Data Fusion*, 1996, vol. 124, pp. 325–345.
- [36] M. Narasimhan and J. Bilmes, "A supermodular-submodular procedure with applications to discriminative structure learning," in *Proc. 21st Conf. UAI*, 2005, pp. 404–412.

- [37] N. Oliver and E. Horvitz, "Selective perception policies for guiding sensing and computation in multimodal systems: A comparative analysis," in *Proc. 5th Int. Conf. Multimodal Interaction*, 2003, pp. 36–43.
- [38] N. Oliver and E. Horvitz, "S-seer: Selective perception in a multimodal office activity recognition system," in *Proc. 1st Int. Workshop MLMI*, 2004, pp. 122–135.
- [39] P. Pudil, J. Novovi, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognit. Lett.*, vol. 15, no. 11, pp. 1119–1125, Nov. 1994.
- [40] R. Rimey and C. Brown, "Control of selective perception using Bayes nets and decision theory," *Int. J. Comput. Vis.*, vol. 12, no. 2/3, pp. 173–207, Apr. 1994.
- [41] R. D. Rimey and C. M. Brown, "Task-specific utility in a general Bayes net vision system," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 1992, pp. 142–147.
- [42] P. Somol, P. Pudil, and J. Kittler, "Fast branch & bound algorithms for optimal feature selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 7, pp. 900–912, Jul. 2004.
- [43] M. Sviridenko, "A note on maximizing a submodular set function subject to knapsack constraint," *Oper. Res. Lett.*, vol. 32, no. 1, pp. 41–43, Jan. 2004.
- [44] L. van der Gaag and M. Wessels, "Selective evidence gathering for diagnostic belief networks," *AISB Q.*, vol. 86, pp. 23–34, 1993.
- [45] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based sensor selection heuristic for target localization," in *Proc. 3rd Int. Symp. IPSN*, 2004, pp. 36–45.
- [46] A. Whitney, "A direct method of non parametric measurement selection," *IEEE Trans. Comput.*, vol. C-20, no. 9, pp. 1100–1103, Sep. 1971.
- [47] H. Wu and A. Cameron, "A Bayesian decision theoretic approach for adaptive goal-directed sensing," in *Proc. ICCV*, 1990, pp. 563–567.
- [48] Y. Zhang, Q. Ji, and C. Looney, "Active information fusion for decision making under uncertainty," in *Proc. Int. Conf. Inf. Fusion*, 2002, vol. 1, pp. 643–650.
- [49] A. X. Zheng, I. Rish, and A. Beygelzimer, "Efficient test selection in active diagnosis via entropy approximation," in *Proc. UAI*, 2005, p. 675.



Wenhui Liao received the Ph.D. degree from the Rensselaer Polytechnic Institute, Troy, NY, in 2006.

She is currently a Research Scientist with the R&D of Thomson-Reuters Corporation, Eagan, MN. Her areas of research include probabilistic graphical models, information fusion, computer vision, and natural language processing.



Qiang Ji (S'92–M'98–SM'04) received the Ph.D. degree in electrical engineering from the University of Washington, Seattle.

He is currently an Associate Professor with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute (RPI), Troy, NY. He currently also serves as a program director at the National Science Foundation. Prior to joining RPI in 2001, he was an Assistant Professor with the Department of Computer Science, University of Nevada, Reno. He also held research and visiting positions with the Beckman Institute, University of Illinois at Urbana–Champaign; the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA; and the U.S. Air Force Research Laboratory. He currently serves as the Director of the Intelligent Systems Laboratory, RPI. His research interests include computer vision, pattern recognition, and probabilistic graphical models. He has published over 100 papers in peer-reviewed journals and conferences. His research has been supported by major governmental agencies including NSF, NIH, DARPA, ONR, ARO, and AFOSR as well as by major companies including Honda and Boeing.

Prof. Ji is an Editor of several computer vision and pattern recognition journals. He has served as a Program Committee Member, Area Chair, and Program Chair in numerous international conferences/workshops.



William A. Wallace (M'90–SM'96–F'02) received the B.Ch.E. degree from the Illinois Institute of Technology, Chicago, in 1956, and the M.S. and Ph.D. degrees in management science from the Rensselaer Polytechnic Institute, Troy, NY, in 1961 and 1965, respectively.

He is currently a Professor with the Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, with joint appointments with the Department of Civil and Environmental Engineering and the Department of Cognitive Science, Rensselaer Polytechnic Institute, where he is also currently the Director of the Center for Infrastructure and Transportation Studies.

Prof. Wallace was the recipient of the International Emergency Management and Engineering Conference Award for Outstanding Long-Term Dedication to the Field of Emergency Management, The Institute of Electrical and Electronics Engineers Third Millennium Medal, and the 2004 INFORMS President's Award for a work that advances the welfare of society.



Efficient non-myopic value-of-information computation for influence diagrams

Wenhui Liao^{a,*}, Qiang Ji^b

^a Research & Development, Thomson-Reuters Corporation, 610 Opperman Drive, Eagan, MN 55123, USA

^b Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA

ARTICLE INFO

Article history:

Received 4 January 2007

Received in revised form 17 April 2008

Accepted 21 April 2008

Available online 29 April 2008

Keywords:

Value-of-information

Influence diagrams

Decision making

Central-limit theorem

Stress modeling

ABSTRACT

In an influence diagram (ID), value-of-information (VOI) is defined as the difference between the maximum expected utilities with and without knowing the outcome of an uncertainty variable prior to making a decision. It is widely used as a sensitivity analysis technique to rate the usefulness of various information sources, and to decide whether pieces of evidence are worth acquisition before actually using them. However, due to the exponential time complexity of exactly computing VOI of multiple information sources, decision analysts and expert-system designers focus on the myopic VOI, which assumes observing only one information source, even though several information sources are available. In this paper, we present an approximate algorithm to compute non-myopic VOI efficiently by utilizing the central-limit theorem. The proposed method overcomes several limitations in the existing work. In addition, a partitioning procedure based on the d -separation concept is proposed to further improve the computational complexity of the proposed algorithm. Both the experiments with synthetic data and the experiments with real data from a real-world application demonstrate that the proposed algorithm can approximate the true non-myopic VOI well even with a small number of observations. The accuracy and efficiency of the algorithm makes it feasible in various applications where efficiently evaluating a large amount of information sources is necessary.

© 2008 Elsevier Inc. All rights reserved.

1. Introduction

In a wide range of decision-making problems, a common scenario is that a decision maker must decide whether some information is worth collecting, and what information should be acquired first given several information sources available. Each set of information sources is usually evaluated by value-of-information (VOI). VOI is a quantitative measure of the value of knowing the outcome of the information source(s) prior to making a decision. In other words, it is quantified as the difference in value achievable with or without knowing the information sources in a decision-making problem.

Generally, VOI analysis is one of the most useful sensitivity analysis techniques for decision analysis [23,25]. VOI analysis evaluates the benefit of collecting additional information in a specific decision-making context [27]. General VOI analyses usually require three key elements: (1) A set of available actions and information collection strategies; (2) A model connecting the actions and the related uncertainty variables within the context of the decision; and (3) values for the decision outcomes. The methods of VOI analysis could be quite different when different models are used.

In this paper, we consider VOI analysis in decision problems modeled by influence diagrams. Influence diagrams were introduced by Howard and Matheson in 1981 [13] and have been widely used as a knowledge representation framework

* Corresponding author. Tel.: +1 5189610457.

E-mail addresses: wenhui.liao@thomsonreuters.com (W. Liao), jiq@rpi.edu (Q. Ji).

to facilitate decision making and probability inference under uncertainty. An ID uses a graphical representation to capture the three diverse sources of knowledge in decision making: conditional relationships about how events influence each other in the decision domain; informational relationships about what action sequences are feasible in any given set of circumstances; and functional relationships about how desirable the consequences are [21]. An ID can systematically model all the relevant random variables and decision variables in a compact graphical model.

In the past several years, a few methods have been proposed to compute VOI in IDs. Ezawa [8] introduces some basic concepts about VOI and evidence propagation in IDs. Dittmer and Jensen [7] present a method for calculating myopic VOI in IDs based on the strong junction tree framework [15]. Shachter [25] further improves this method by enhancing the strong junction tree as well as developing methods for reusing the original tree in order to perform multiple VOI calculations. Zhang et al. [28] present an algorithm to speed up the VOI computation by making use of the intermediate computation results, which are obtained when computing the optimal expected value of the original ID without the observations from the information sources. Instead of computing VOI directly, [22] describe a procedure to identify a partial order over variables in terms of their VOIs based on the topological relationships among variables in the ID. However, all these papers only focus on computing myopic VOI, which is based on two assumptions: (1) “No competition:” each information source is evaluated in isolation, as if it were the only source available for the entire decision; (2) “One-step horizon:” the decision maker will act immediately after consulting the source [21]. These assumptions result in a myopic policy: every time, the decision maker evaluates the VOI of each information source one by one, and chooses the one with the largest VOI. Then the observations are collected from the selected information sources, the probabilities are updated, and all the remaining information sources are to be reevaluated again, and a similar procedure repeats.

Obviously, the assumptions are not always reasonable in some decision circumstances. Usually, the decision maker will not act after acquiring only one information source. Also, although a single information source may have low VOI and is not worth acquisition compared to its cost, several information sources used together may have high VOI compared to their combined cost. In this case, by only evaluating myopic VOI, the conclusion will be not to collect such information, which is not optimal since its usage together with other information sources can lead to high value for the decision maker. Therefore, given these limitations in myopic VOI, it is necessary to compute non-myopic VOI.

Non-myopic VOI respects the fact that the decision maker may observe more than one piece of information before acting, thus requires the consideration of any possible ordered sequence of observations given a set of information sources. Unfortunately, the number of the sequences grows exponentially as the number of available information sources increases, and thus it is usually too cumbersome to compute non-myopic VOI for any practical use, and this is why the before mentioned work only focuses on myopic VOI. Given these facts, an approximate computation of non-myopic VOI is necessary to make it feasible in practical applications. To the best of our knowledge, [11] are the only ones who proposed a solution to this problem. In their approach, the central-limit theorem is applied to approximately compute non-myopic VOI in a special type of ID for the diagnosis problem, where only one decision node exists. Certain assumptions are required in their method: (1) all the random nodes and decision nodes in the ID are required to be binary; (2) the information sources are conditionally independent from each other given the hypothesis node, which is the node associated with the decision node and utility node.

Motivated by the method of Heckerman et al., we extend this method to more general cases¹: (1) all the random nodes can have multiple states and the decision node can have multiple rules (alternatives); (2) the information sources can be dependent given the hypothesis node; and (3) the ID can have a more general structure. But same as Heckerman et al.’s method, we only discuss the VOI computation in terms of IDs that have only one decision node. This decision node shares only one utility node with another chance node. With the proposed algorithm, non-myopic VOI can be efficiently approximated. In order to validate the performance of the proposed algorithm, we not only perform the experiments based on the synthetic data for various types of IDs, but also provide a real-world application with real data.

Because of the efficiency and accuracy of the proposed method, we believe that it can be widely used to choose the optimal set of available information sources for a wide range of applications. No matter what selection strategies people use to choose an optimal set, such as greedy approaches, heuristic searching algorithms, or brute-force methods, the proposed method can be utilized to evaluate any information set efficiently in order to speed up the selection procedure.

The following sections are organized as follows. Section 2 presents a brief introduction to influence diagrams. The detail of the algorithm is described in Section 3. Section 4 discusses the experimental results based on synthetic data. And a real application is demonstrated in Section 5. Finally, Section 6 gives the conclusion and some suggestions for future work.

2. Influence diagrams

An influence diagram (ID) is a graphical representation of a decision-making problem under uncertainty. Its knowledge representation can be viewed through three hierarchical levels, namely, relational, functional, and numerical. At the relational level, an ID represents the relationships between different variables through an acyclic directed graph consisting of various node types and directed arcs. The functional level specifies the interrelationships between various node types and defines the corresponding conditional probability distributions. Finally, the numerical level specifies the actual numbers associated with the probability distributions and utility values [6].

¹ A brief version of this extension can be found in [18].

Specifically, an ID includes three types of nodes: decision, chance (random), and value (utility) nodes. Decision nodes, usually drawn as rectangles, indicate the decisions to be made and their set of possible alternative values. Chance nodes, usually drawn as circles/ellipses, represent uncertain variables that are relevant to the decision problem. They are similar to the nodes in Bayesian networks [14], and are associated with conditional probability tables (CPTs). Value nodes, usually drawn as diamonds, are associated with utility functions to represent the utility of each possible combination of the outcomes of the parent node. The arcs connecting different types of nodes have different meanings. An arc between two chance nodes represents probabilistic dependence, while an arc from a decision node to a chance node represents functional dependence, which means the actions associated with the decision node affect the outcome of the chance node. An arc between two decision nodes implies time precedence, while an arc from a chance node to a decision node is informational, i.e., it shows which variable will be known to the decision maker before a decision is made [21]. An arc pointing to a utility node represents value influence, which indicates that the parents of the utility node are those that directly affect its utility. Fig. 1 illustrates these arcs and gives corresponding interpretations.

Most IDs assume a precedence ordering of the decision nodes. A regular ID assumes that there is a directed path containing all decision nodes; a no-forgetting ID assumes that each decision node and its parents are also parents of the successive decision nodes; and a stepwise decomposable ID assumes that the parents of each decision node divide the ID into two separate fractions. In this paper, we consider IDs that have only one decision node, i.e., ignoring all previous decisions. The goal of ID modeling is to choose an optimal policy that maximizes the overall expected utility. A policy is a sequence of decision rules where each rule corresponds to one decision node. Mathematically, if there is only one decision node in an ID and assuming additive decomposition of the utility functions, the expected utility under a decision rule d given any available evidence e , denoted by $EU(d|e)$, can be defined as follows:

$$EU(d|e) = \sum_{i=1}^n \sum_{X_i} p(X_i|e, d)u_i(X_i, d), \tag{1}$$

where u_i is the utility function over the domain $X_i \cup \{D\}$. For example, X_i could be the parents of the utility node that u_i is associated with. To evaluate an ID is to find an optimal policy as well as to compute its optimal expected utility [24,26]. More detail about IDs can be found in [17,14].

Generally, the advantages of an ID can be summarized by its compact and intuitive formulation, its easy numerical assessment, and its effective graphical representation of dependence between variables for modeling decision making under uncertainty. These benefits make ID a widely used tool to model and solve complex decision problems in recent years.

3. Approximate VOI computation

3.1. Value of information

The VOI of a set of information sources is defined as the difference between the maximum expected utilities with and without the information sources [17]. VOI can be used to rate the usefulness of various information sources and to decide whether pieces of evidence are worth acquisition before actually using the information sources [21].

We discuss the VOI computation in terms of IDs that have only one decision node. This decision node shares only one utility node with another chance node, as shown in Fig. 2. And the decision node and the chance node are assumed to be independent. In the ID, the chance node Θ , named as hypothesis node, represents a mutually exclusive and exhaustive set of possible hypotheses $\theta_1, \theta_2, \dots, \theta_n$; the decision node D represents a set of possible alternatives d_1, d_2, \dots, d_q ; the utility node U represents the utility of the decision maker, which depends on the outcome of Θ and D ; and the chance nodes O_1, \dots, O_n represent possible observations from all kinds of information sources about the true state of Θ . And each O_i may have multiple states. Let $O = \{O_1, \dots, O_n\}$, the VOI of O , $VOI(O)$, w.r.t. the decision node D , can be defined as follows:

$$VOI(O) = EU(O) - EU(\bar{O}), \tag{2}$$

$$EU(O) = \sum_{o \in O} p(o) \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i|o)u(\theta_i, d_j), \tag{3}$$

$$EU(\bar{O}) = \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i)u(\theta_i, d_j), \tag{4}$$

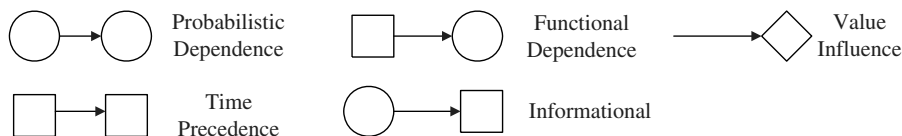


Fig. 1. Interpretations of arcs in an ID, where circles represent chance (random) nodes, rectangles for decision nodes, and diamonds for value (utility) nodes.

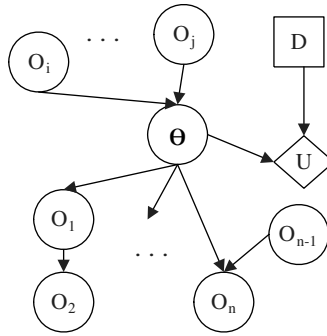


Fig. 2. An ID example for non-myopic VOI computation. θ is the hypothesis node, D is the decision node, and U is the utility node. O_i represents possible observations from an information source. There could be hidden nodes between θ and O_i .

where $u(\cdot)$ denotes the utility function associated with the utility node U , $EU(O)$ denotes the expected utility to the decision maker if O were observed, while $EU(\bar{O})$ denotes the expected utility to the decision maker without observing O . Here the cost of collecting information from the information sources is not included; thus, the VOI can also be called perfect VOI [11]. The net VOI is the difference between the perfect VOI and the cost of collecting information [12]. Since after calculating the perfect VOI, the computation of the net VOI is just a subtraction of cost, we focus on the perfect VOI in the subsequent sections.

As shown in Eq. (2), to compute $VOI(O)$, it is necessary to compute $EU(O)$ and $EU(\bar{O})$ respectively. Obviously, $EU(\bar{O})$ is easier to compute, whereas directly computing $EU(O)$ could be cumbersome. If the decision maker has the option to observe a subset of observations $\{O_1, \dots, O_n\}$ and each O_i has m possible values, then there are m^n possible instantiations of the observations in this set. Thus, to compute $EU(O)$, there are m^n inferences to be performed. In other words, the time complexity of computing VOI is exponential. It becomes infeasible to compute $VOI(O)$ when n is not small.

The key to computing $VOI(O)$ efficiently is to compute $EU(O)$, which can be rewritten as follows:

$$EU(O) = \sum_{o \in O} p(o) \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i | o) u(\theta_i, d_j) = \sum_{o \in O} \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(o) p(\theta_i | o) u(\theta_i, d_j) = \sum_{o \in O} \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i) p(o | \theta_i) u(\theta_i, d_j). \quad (5)$$

It is noticed that each instantiation of O corresponds to a specific optimal action for the decision node D . We define the decision function $\delta : O \rightarrow D$, which maps an instantiation of O into a decision in D . For example, $\delta(o) = d_k$ indicates when the observation is o , the corresponding optimal decision is d_k , $d_k = \arg \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i | o) u(\theta_i, d_j)$. Therefore we can divide all the instantiations of O into several subsets, where the optimal action is the same for those instantiations in the same subset. Specifically, if D has q decision rules, $\{d_1, \dots, d_q\}$, all the instantiations of O can be divided into q subsets, $o_{d_1}, o_{d_2}, \dots, o_{d_q}$, where $o_{d_k} = \{o \in O | \delta(o) = d_k\}$. Fig. 3 illustrates the relationships between each instantiation and the q subsets. Thus, from Eq. (5), $EU(O)$ can be further derived as follows:

$$EU(O) = \sum_{\theta_i \in \Theta} p(\theta_i) \sum_{k=1}^q \sum_{o \in o_{d_k}} p(o | \theta_i) u(\theta_i, d_k). \quad (6)$$

In the next several sections, we show how to compute $EU(O)$ efficiently.

3.2. Decision boundaries

In Eq. (6), the difficult part is to compute $\sum_{o \in o_{d_k}} p(o | \theta_i)$ because the size of the set o_{d_k} could be very large based on the previous analysis. In order to compute it efficiently, it is necessary to know how to divide all the instantiations of O into the q subsets. We first focus on the case that Θ has only two states, θ_1, θ_2 , and then extend it to the general case in Section 3.4.

Based on the definition, the expected utility of taking the action d_k is $EU(d_k) = p(\theta_1) * u_{1k} + p(\theta_2) * u_{2k}$, where $u_{1k} = u(\theta_1, d_k)$, and $u_{2k} = u(\theta_2, d_k)$. We can sort the index of all the decision rules based on the utility functions, such that $u_{1k} > u_{1j}$ and $u_{2k} < u_{2j}$ for $k < j$. Fig. 4 gives an example of the utility function $u(\theta, D)$. As shown in the figure, as k increases, u_{1k} decreases and u_{2k} increases. If there is an action d_i that cannot be sorted according to this criterion, it is either

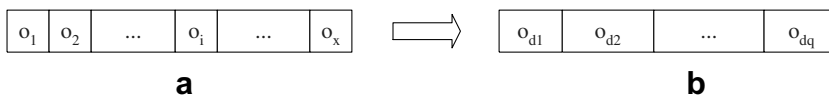


Fig. 3. (a) Each o_i corresponds to an instantiation; (b) all the instantiations can be divided into q subsets, where each instantiation in the set o_{d_i} corresponds to the optimal decision d_i .

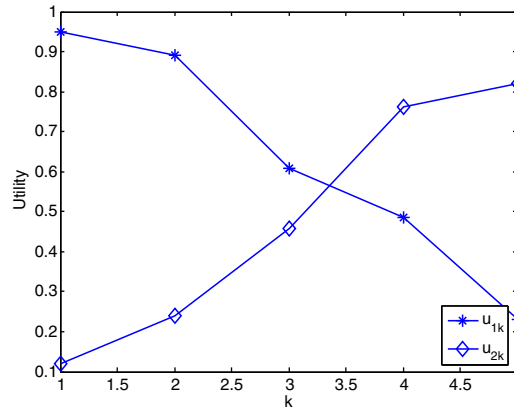


Fig. 4. An example of the utility function $U(\theta, D)$.

dominated by another action, or it dominates another action. (If $u(d_i, \theta)$ is always larger than $u(d_j, \theta)$, no matter what state of θ is, we say d_i dominates d_j). Then the dominated action can be removed from the set of possible actions, without changing the optimal policy.

Proposition 1. Let $r_{jk} = \frac{u_{2j} - u_{2k}}{u_{1k} - u_{1j} + u_{2j} - u_{2k}}$, $p_{kl}^* = \max_{k < j \leq q} r_{jk}$, and $p_{ku}^* = \min_{1 \leq j < k} r_{jk}$, then d_k is the optimal action if and only if $p_{kl}^* \leq p(\theta_1) \leq p_{ku}^*$. In addition, $p_{ql}^* = 0$ and $p_{1u}^* = 1$. (Here k is the index of an action.)

Proof. see Appendix. \square

Proposition 1 presents that if the probability of θ being θ_1 is between p_{kl}^* and p_{ku}^* , d_k is the optimal decision. From this, we can further derive Proposition 2.

Proposition 2

$$\sum_{\theta \in \theta_{d_k}} p(\theta) = p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^*). \tag{7}$$

Proof. see Appendix. \square

The Proof of Proposition 2 establishes Eq. (7) by showing that both sides of this equation express the probability that d_k is the optimal decision for θ_1 . Based on Proposition 2, we can get the following corollary.

Corollary 1

$$\sum_{\theta \in \theta_{d_k}} p(\theta | \theta_1) = p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_1), \tag{8}$$

$$\sum_{\theta \in \theta_{d_k}} p(\theta | \theta_2) = p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_2). \tag{9}$$

The equations in Corollary 1 indicate the probability that the decision maker will take the optimal decision d_k after observing new evidence, given the situation that the state of θ is θ_i before collecting the evidence.

Based on Corollary 1, the problem of computing $\sum_{\theta \in \theta_{d_k}} p(\theta | \theta_i)$, $i = 1, 2$, (from Eq. (6)) transfers to the problem of computing $p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_i)$, which is the topic of the next section. We will focus on $p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_1)$ only because the procedure of computing $p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_2)$ is similar.

3.3. Approximation with central-limit theorem

3.3.1. A partitioning procedure

To compute $p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_1)$, one way is to treat $p(\theta_1 | \theta)$ as a random variable. If the probability density function of this variable is known, it will be easy to compute $p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_1)$. However, it is hard to get such a probability density function directly. But we notice that $p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_1) = p\left(\frac{p_{kl}^*}{1-p_{kl}^*} \leq \frac{p(\theta_1 | \theta)}{p(\theta_2 | \theta)} \leq \frac{p_{ku}^*}{1-p_{ku}^*} | \theta_1\right)$. Based on the transformation property between a random variable and its function [2], it is straightforward that $p(p_{kl}^* \leq p(\theta_1 | \theta) \leq p_{ku}^* | \theta_1) = p\left(\frac{p_{kl}^*}{1-p_{kl}^*} \leq \frac{p(\theta_1 | \theta)}{p(\theta_2 | \theta)} \leq \frac{p_{ku}^*}{1-p_{ku}^*} | \theta_1\right)$.

Let us take a closer look at $\frac{p(\theta_1 | \theta)}{p(\theta_2 | \theta)}$ because it is critical in the approximate algorithm.

If all the O_i nodes are conditionally independent from each other given Θ , based on the chain rule:

$$\frac{p(\theta_1|O)}{p(\theta_2|O)} = \frac{p(O_1|\theta_1)}{p(O_1|\theta_2)} \dots \frac{p(O_n|\theta_1)}{p(O_n|\theta_2)} \frac{p(\theta_1)}{p(\theta_2)} \tag{10}$$

Usually some O_i s may not be conditionally independent given Θ . We will show that $\frac{p(\theta_1|O)}{p(\theta_2|O)}$ is approximately distributed as a log-normal random variable. However, in order to prove it, it is necessary to obtain a format similar to Eq. (10) even when O_i s are not conditionally independent. We thus propose a partitioning procedure to partition O into several groups based on the principle of d -separation [21], where the nodes in one group are conditionally independent from the nodes in other groups. This procedure consists of three steps.

- (1) Decide whether two nodes, O_i, O_j , are conditionally independent given Θ by exploring the ID structure based on four rules: (i) if there is a **directed** path between O_i and O_j without passing Θ , O_i and O_j are dependent; (ii) if both O_i and O_j are the ancestors of Θ , O_i and O_j are dependent given Θ ; (iii) after removing the links to and from Θ from the original ID, if O_i and O_j have common ancestors, or O_i is O_j 's ancestor, or vice versa, then O_i and O_j are dependent; and (iv) in all the other cases, O_i and O_j are conditionally independent given Θ .
- (2) Build an undirected graph to model the relationships between the nodes. In such a graph, each vertex represents an O_i node, and each edge between two vertices indicates that the two corresponding nodes are dependent according to the rules in Step 1.
- (3) Partition the graph into disjoint connected subgraphs. A depth first search (DFS) algorithm [4] is used to partition the graph into several *connected components* (disjoint connected subgraphs) so that each component is disconnected from other components. The nodes in each connected component are conditionally independent from the nodes in any other connected components. Therefore, each connected component corresponds to one group.

For example, for the ID in Fig. 5a, with the partitioning procedure, the O_i nodes can be divided into five groups, $\{O_1, O_2\}$, $\{O_3, O_4, O_5\}$, $\{O_6\}$, $\{O_7\}$, and $\{O_8, O_9\}$. Fig. 5b shows the graph built by the partitioning procedure.

3.3.2. Central-limit theorem

Generally, with the partition procedure presented in the previous subsection, O can be automatically divided into several sets, named $O^{s_1}, O^{s_2}, \dots, O^{s_g}$, where g is the overall number of the groups. Thus, Eq. (10) can be modified as follows:

$$\frac{p(\theta_1|O)}{p(\theta_2|O)} = \frac{p(O^{s_1}|\theta_1)}{p(O^{s_1}|\theta_2)} \dots \frac{p(O^{s_g}|\theta_1)}{p(O^{s_g}|\theta_2)} \frac{p(\theta_1)}{p(\theta_2)} \Rightarrow \ln \frac{p(\theta_1|O)}{p(\theta_2|O)} = \sum_{i=1}^g \ln \frac{p(O^{s_i}|\theta_1)}{p(O^{s_i}|\theta_2)} + \ln \frac{p(\theta_1)}{p(\theta_2)} \Rightarrow \ln \phi = \sum_{i=1}^g w_i + c,$$

where $\phi = \frac{p(\theta_1|O)}{p(\theta_2|O)}$, $w_i = \ln \frac{p(O^{s_i}|\theta_1)}{p(O^{s_i}|\theta_2)}$, $c = \ln \frac{p(\theta_1)}{p(\theta_2)}$. (11)

In the above equation, c can be regarded as a constant reflecting the state of Θ before any new observation is obtained and any new decision is taken. Here, we assume $p(\theta_2|O)$, $p(O^{s_i}|\theta_2)$, and $p(\theta_2)$ are not equal to 0.

Let $W = \sum_{i=1}^g w_i$ be the sum of w_i . Following [11], we use the central-limit theorem to approximate W . The central-limit theorem [9] states that the sum of independent variables approaches a Gaussian distribution when the number of variables becomes large. Also, the expectation and variance of the sum is the sum of the expectation and variance of each individual random variable. Thus, regarding each w_i as an independent variable, W then follows a Gaussian distribution. Then, based on Eq. (11), ϕ will be a log-normal distribution. For a random variable X , if $\ln(X)$ has a Gaussian distribution, we say X has a log-normal distribution. The probability density function is: $p(x) = \frac{1}{S\sqrt{2\pi x}} e^{-(\ln x - M)^2 / (2S^2)}$, denoted as $X \sim \text{LogN}(M, S^2)$ [5], where M and S are the mean and standard deviation of the variable's logarithm [1]. In order to assess the parameters (mean and var-

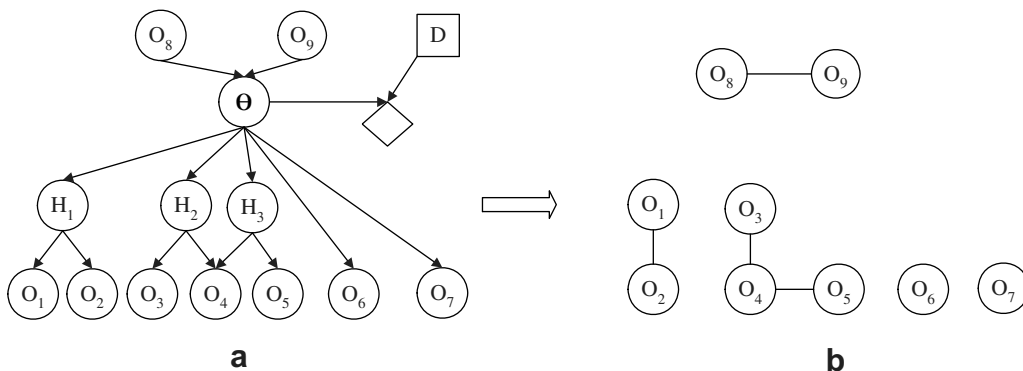


Fig. 5. (a) An ID example; (b) the graph built by the partitioning procedure.

iance) of the log-normal distribution, we need to compute the mean and the variance of each w_i . The computational process is shown as follows.

Assume O^{S_i} has r_i instantiations, $\{o_1^{S_i}, \dots, o_{r_i}^{S_i}\}$, where r_i is the product of the number of the states for each node in the group O^{S_i} , e.g., if $O^{S_i} = \{O_1, O_2\}$, and both O_1 and O_2 have three states, then $r_i = 3 * 3 = 9$. Table 1 gives the value and the probability distribution for each w_i :

Based on the table, the expected value μ , and the variance σ^2 for each w_i can be computed as follows:

$$\mu(w_i|\theta_1) = \sum_{j=1}^{r_i} p(o_j^{S_i}|\theta_1) \ln \frac{p(o_j^{S_i}|\theta_1)}{p(o_j^{S_i}|\theta_2)}, \tag{12}$$

$$\sigma^2(w_i|\theta_1) = \sum_{j=1}^{r_i} p(o_j^{S_i}|\theta_1) \ln^2 \frac{p(o_j^{S_i}|\theta_1)}{p(o_j^{S_i}|\theta_2)} - \mu^2(w_i|\theta_1). \tag{13}$$

By the central-limit theorem, the expected value and the variance of W can be obtained by the following equations:

$$\mu(W|\theta_1) = \sum_{i=1}^g \mu(w_i|\theta_1), \tag{14}$$

$$\sigma^2(W|\theta_1) = \sum_{i=1}^g \sigma^2(w_i|\theta_1). \tag{15}$$

Therefore, based on Eq. (11), for $W \sim N(\mu(W|\theta_1), \sigma^2(W|\theta_1))$, we have $\phi \sim \text{LogN}(\mu(W|\theta_1) + c, \sigma^2(W|\theta_1))$, where LogN denotes the log-normal distribution. After getting the probability distribution function and the function parameters for ϕ in Eq. (11), we are ready to assess the non-myopic VOI.

Before we go to the next section, we first analyze the computational steps involved in computing the parameters for the log-normal distribution, which is the most time-consuming part in the algorithm. Based on Eqs. (12) and (14), the overall number of the computational steps is $4 \sum_{i=1}^g r_i + 2g$. We will show that this number is much smaller than the overall number of the computational steps in the exact computational method during the algorithm analysis in Section 3.5.

3.3.3. Approximate non-myopic value-of-information

Based on Proposition 1 in Section 3.2, we know that d_k is the optimal action with the probability $p(p_{kl}^* \leq p(\theta_1|o) \leq p_{ku}^*)$, which is equivalent to $p\left(\frac{p_{kl}}{1-p_{kl}} \leq \phi \leq \frac{p_{ku}}{1-p_{ku}}\right)$ as shown in Section 3.3.1. Let $\phi_{kl}^* = \frac{p_{kl}}{1-p_{kl}}$, and $\phi_{ku}^* = \frac{p_{ku}}{1-p_{ku}}$, thus, d_k is the optimal decision if and only if $\phi_{kl}^* \leq \phi \leq \phi_{ku}^*$. Then, based on Corollary 1 in Section 3.2, the following equation stands:

$$\sum_{o \in o_{d_k}} p(o|\theta_1) = p(\phi_{kl}^* \leq \phi \leq \phi_{ku}^*|\theta_1). \tag{16}$$

Furthermore, from Section 3.3.2, we know that $\phi \sim \text{LogN}(\mu(W|\theta_1) + c, \sigma^2(W|\theta_1))$, thus,

$$p(\phi_{kl}^* \leq \phi \leq \phi_{ku}^*|\theta_1) = \frac{1}{\sigma(W|\theta_1)\sqrt{2\pi x}} \int_{\phi_{kl}^*}^{\phi_{ku}^*} e^{-\frac{(\ln x - \mu(W|\theta_1) - c)^2}{2\sigma^2(W|\theta_1)}} dx, \tag{17}$$

$p(\phi_{kl}^* \leq \phi \leq \phi_{ku}^*|\theta_2)$ can be computed in the same way by replacing θ_1 with θ_2 in the previous equations.

Therefore, VOI can be approximated by combining Eqs. (2), (6), (16), and (17). Fig. 6 shows the key equations of the algorithm when Θ has only two states. In summary, to approximate VOI(O) efficiently, the key is to compute $EU(O)$, which leads to an approximation of $\sum_{o \in o_{d_k}} p(o|\theta_1)$ with the log-normal distribution by exploiting the central-limit theorem and the decision boundaries.

3.4. Generalization

In the previous algorithm, the node Θ only allows two states, although the other random nodes and the decision node can be multiple states. However, in real-world applications, Θ may have more than two states. In this section, we extend the algorithm to the case that Θ can have several states too. Assume Θ has h states, $\theta_1, \dots, \theta_h$, and still, d has q rules, d_1, \dots, d_q , similarly to Eq. (11), we have the following equations:

Table 1
The probability distribution of w_i

w_i	$p(w_i \theta_1)$	$p(w_i \theta_2)$
$\ln \frac{p(o_1^{S_i} \theta_1)}{p(o_1^{S_i} \theta_2)}$	$p(o_1^{S_i} \theta_1)$	$p(o_1^{S_i} \theta_2)$
\dots	\dots	\dots
$\ln \frac{p(o_{r_i}^{S_i} \theta_1)}{p(o_{r_i}^{S_i} \theta_2)}$	$p(o_{r_i}^{S_i} \theta_1)$	$p(o_{r_i}^{S_i} \theta_2)$

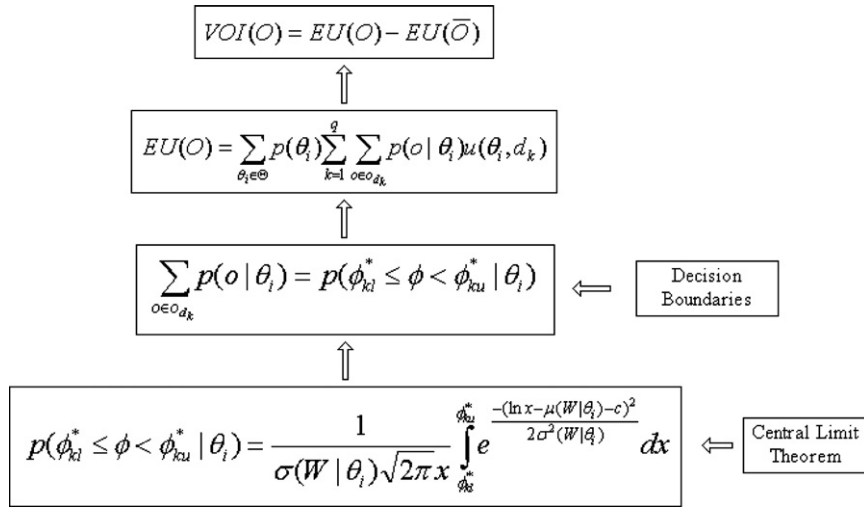


Fig. 6. The key equations to approximate VOI when θ has only two states, D has multiple rules, and the other nodes have multiple states.

$$\frac{p(\theta_i|O)}{p(\theta_h|O)} = \frac{p(O^{s_1}|\theta_i)}{p(O^{s_1}|\theta_h)} \dots \frac{p(O^{s_g}|\theta_i)}{p(O^{s_g}|\theta_h)} \frac{p(\theta_i)}{p(\theta_h)}, \quad i \neq h \Rightarrow \ln \frac{p(\theta_i|O)}{p(\theta_h|O)} = \sum_{k=1}^g \ln \frac{p(O^{s_k}|\theta_i)}{p(O^{s_k}|\theta_h)} + \ln \frac{p(\theta_i)}{p(\theta_h)} \Rightarrow \ln \phi_i$$

$$= \sum_{k=1}^g w_k^i + c_i, \quad \text{where } \phi_i = \frac{p(\theta_i|O)}{p(\theta_h|O)}, \quad w_k^i = \ln \frac{p(O^{s_k}|\theta_i)}{p(O^{s_k}|\theta_h)}, \quad c_i = \ln \frac{p(\theta_i)}{p(\theta_h)}. \quad (18)$$

Let $W_i = \sum_{k=1}^g w_k^i$, $i \neq h$, W_i still has a Gaussian distribution. Here, we assume $p(\theta_h|O)$, $p(O^{s_k}|\theta_h)$, and $p(\theta_h)$ are not equal to 0. The similar method in Section 3.3 can be used to compute the variance and the mean. Specifically, for the new defined w_k^i in the above equation, Table 1 can be modified as follows (see Table 2).

Thus, we get the following equations:

$$\mu(w_k^i|\theta_j) = \sum_{l=1}^{r_k} p(o_l^{s_k}|\theta_j) \ln \frac{p(o_l^{s_k}|\theta_i)}{p(o_l^{s_k}|\theta_h)}, \quad 1 \leq i < h, \quad 1 \leq j \leq h, \quad 1 \leq k \leq g, \quad (19)$$

$$\sigma^2(w_k^i|\theta_j) = \sum_{l=1}^{r_k} p(o_l^{s_k}|\theta_j) \ln^2 \frac{p(o_l^{s_k}|\theta_i)}{p(o_l^{s_k}|\theta_h)} - \mu^2(w_k^i|\theta_j). \quad (20)$$

Similar to Eq. (14), the expected value and the variance of W_i can be obtained as we see here:

$$\mu(W_i|\theta_j) = \sum_{k=1}^g \mu(w_k^i|\theta_j), \quad 1 \leq i < h, \quad 1 \leq j \leq h, \quad (21)$$

$$\sigma^2(W_i|\theta_j) = \sum_{k=1}^g \sigma^2(w_k^i|\theta_j). \quad (22)$$

Accordingly, ϕ_i follows the log-normal distribution with $S_{ij} = \sigma(W_i|\theta_j)$ and $M_{ij} = \mu(W_i|\theta_j) + c_i$. We denote the probability density function of ϕ_i given θ_j as $f_{\theta_j}(\phi_i)$. Eqs. (19) and (21) show that the overall number of the computational steps to assess the parameters for the log-normal distributions is $4h \sum_{k=1}^g r_k + 2h(h-1)g$ when $h > 2$.

Even though $f_{\theta_j}(\phi_i)$ can be easily obtained, it is still necessary to get the decision boundaries for each optimal decision in order to efficiently compute $\sum_{o \in O_{d_k}} p(o|\theta_j)$. Therefore, a set of linear inequality functions need to be solved when θ has more than two states. For example, if d_k is the optimal action, $EU(d_k)$ must be larger than the expected utility of taking any other action. Based on this, a set of linear inequality functions can be obtained:

Table 2
The probability distribution of w_k^i

w_k^i	$p(w_k^i \theta_1)$...	$p(w_k^i \theta_h)$
$\ln \frac{p(o_1^{s_k} \theta_i)}{p(o_1^{s_k} \theta_h)}$	$p(o_1^{s_k} \theta_1)$...	$p(o_1^{s_k} \theta_h)$
...
$\ln \frac{p(o_{r_k}^{s_k} \theta_i)}{p(o_{r_k}^{s_k} \theta_h)}$	$p(o_{r_k}^{s_k} \theta_1)$...	$p(o_{r_k}^{s_k} \theta_h)$

$$\begin{aligned}
 p(\theta_1)u_{1k} + p(\theta_2)u_{2k} + \dots + p(\theta_h)u_{hk} &\geq p(\theta_1)u_{1j} + \dots + p(\theta_h)u_{hj} \\
 &\Rightarrow \frac{u_{1k} - u_{1j} + u_{hj} - u_{hk}}{u_{hj} - u_{hk}} \cdot p(\theta_1) + \dots + \frac{u_{(h-1)k} - u_{(h-1)j} + u_{hj} - u_{hk}}{u_{hj} - u_{hk}} \cdot p(\theta_{h-1}) \geq 1 \\
 &\Rightarrow \frac{u_{1k} - u_{1j}}{u_{hj} - u_{hk}} \cdot \frac{p(\theta_1)}{p(\theta_h)} + \dots + \frac{u_{(h-1)k} - u_{(h-1)j}}{u_{hj} - u_{hk}} \cdot \frac{p(\theta_{h-1})}{p(\theta_h)} \geq 1.
 \end{aligned} \tag{23}$$

We assume $u_{hj} - u_{hk} > 0$; otherwise, “ \geq ” is changed to “ \leq ” in the last inequality.

Let A_k be the solution region of the above linear inequalities, then

$$\sum_{\theta \in \Theta_k} p(\theta_j) = \int_{A_k} f_{\theta_j}(\phi_1) \dots f_{\theta_j}(\phi_{h-1}) dA_k, \quad 1 \leq j \leq h, \quad 1 \leq k \leq q. \tag{24}$$

The right side of Eq. (24) is an integral over the solution region A_k decided by the linear inequalities. We first demonstrate how to solve the integral when Θ has three states, and then introduce the method for the case that Θ has more than three states.

When Θ has three states, Eq. (23) can be simplified as follows:

$$\begin{aligned}
 p(\theta_1)u_{1k} + p(\theta_2)u_{2k} + p(\theta_3)u_{3k} &\geq p(\theta_1)u_{1j} + p(\theta_2)u_{2j} + p(\theta_3)u_{3j} \Rightarrow \alpha_{1kj} \cdot \frac{p(\theta_1)}{p(\theta_3)} + \alpha_{2kj} \cdot \frac{p(\theta_2)}{p(\theta_3)} \geq 1, \\
 \text{where } \alpha_{1kj} &= \frac{u_{1k} - u_{1j}}{u_{3j} - u_{3k}} \text{ and } \alpha_{2kj} = \frac{u_{2k} - u_{2j}}{u_{3j} - u_{3k}}.
 \end{aligned} \tag{25}$$

In the above, it is assumed that $u_{3j} > u_{3k}$; if $u_{3j} < u_{3k}$, then “ \geq ” is changed to “ \leq ” in the last inequality.

And Eq. (24) can be simplified as follows:

$$\sum_{\theta \in \Theta_k} p(\theta_j) = \int_{A_k} f_{\theta_j}(\phi_1) f_{\theta_j}(\phi_2) dA_k, \quad 1 \leq k \leq q, \quad 1 \leq j \leq 3, \tag{26}$$

A_k is decided by $(q - 1)$ linear inequalities and each inequality has two variables ϕ_1 and ϕ_2 as defined in Eq. (25). We use the following steps to solve this integral when A_k is a finite region.

1. Identify all the lines that define the inequalities and find all the intersection points between any two lines as well as the intersection points between any line and the x (or y) axis.
2. Choose the intersection points that satisfy all the linear inequalities, and use them as vertices to form a polygon.
3. Divide the polygon into several simple regions: Specifically, for each vertex, we generate a line crossing this vertex and parallel to the y -axis. The lines then divide the polygon into several simple regions.
4. Evaluate the integral in each simple region and sum the values together.

An example of the solution region is shown in Fig. 7. In this example, if $\alpha_{1kj} > \alpha_{1kj} (i \neq j)$, then $\alpha_{2kj} > \alpha_{2kj}$ too. Therefore, the solution region can be decided by the intersection points of the lines that are defined by the linear inequalities and the axes. For example, in Fig. 7, A_k is decided by a–d, which are selected from the intersection points $\{(1/\alpha_{1kj}, 0), (0, 1/\alpha_{2kj}), j = 1, \dots, q, j \neq k\}$. Based on [3], the time complexity of solving m linear inequalities with n variables (each inequality only has two variables) is $O(mn \log m + mn^2 \log^2 n)$. In this case, n is 2 and m is $q - 1$.

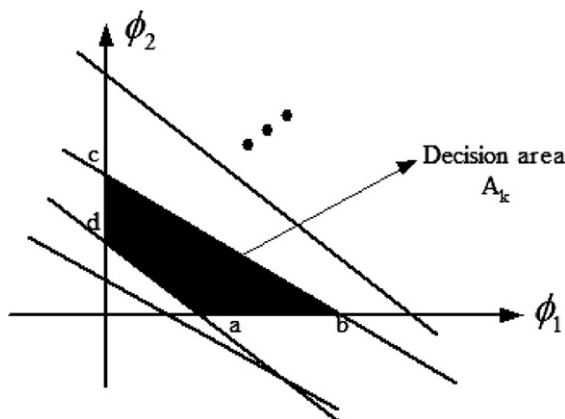


Fig. 7. A solution region of a group of linear inequalities.

When Θ has more than three states, the integral needs to be performed in a high-dimension space (dimension is larger than 2). Therefore, we solve it with Quasi-Monte Carlo integration [10,16], which is a popular method to handle multiple integral. Quasi-Monte Carlo integration picks points based on sequences of quasirandom numbers over some simple domain A'_k which is a superset of A_k , checks whether each point is within A_k , and estimates the area (n -dimensional content) of A_k as the area of A'_k multiplied by the fraction of points falling within A_k . Such a method is implemented by Mathematica [20], which can automatically handle a multiple integral with a region implicitly defined by multiple inequality functions.

Fig. 8 shows the key equations of the algorithm when Θ has multiple states. The main equations are similar to those in Fig. 6. However, since Θ has multiple states, it becomes more complex to obtain the parameters of the log-normal distribution and perform the integration.

3.5. Algorithm analysis

Now, we analyze the computational complexity of the proposed approximation algorithm compared to the exact computational method. For simplicity, assume that the number of the state of each O_i node is m , and there are n nodes in the set O . Assume we only count the time used for computing expected utilities. Then the computational complexity of the exact VOI computational method is approximately hm^n , where h is the number of the state of the Θ node. With the approximation algorithm, the computational complexity is reduced to hm^k , where h is the number of the state of the Θ node, and k is the number of O_i nodes in the maximum group among $\{O^{s_1}, \dots, O^{s_g}\}$. In the best case, if all the O_i nodes are conditionally independent given θ , the time complexity is about linear with respect to m . In the worst case, if all the O_i nodes are dependent, the time complexity is approximately m^n . However, usually, in most real-world applications, k is less than n , thus, the approximate algorithm is expected to be more efficient than the exact computational method, as will be shown in the experiments. For example, for the ID in Fig. 5, $n = 9$, $m = 4$, $h = 3$, and $q = 3$. Then, for the exact computation, the number of computations is around $3 * 4^9 = 786432$, while using the approximate algorithm, the number of computations is only around $3 * 4^3 = 192$.

However, in addition to the cost of computing expected utilities, the approximation algorithm also includes some extra costs: sorting the utility functions (Section 3.2), partitioning the O set (Section 3.3.1), and deciding the decision boundaries (Section 3.2) when θ has two states, or performing the integral when θ has more than two states (Section 3.4). These costs are not included in the above analysis. In general, the extra time in these steps is much less than the time used for computing expected utilities. For example, the time complexity of sorting is $O(q \log(q))$, the time complexity of the partition procedure is $O(|V| + |E|)$ (V is the set of vertex, and E is the set of edges in an ID), and the time complexity in deciding the decision boundaries when θ has two states is $O(q^2)$. When θ has more than two states, deciding the decision boundaries needs additional time. Empirically, it does not affect the overall speed, as will be shown in the experiments. In addition, most steps in computing expected utilities involve performing inferences in an ID, which is usually NP-hard and thus consumes much more time than a step in the procedures of sorting, partitioning, and integrating.

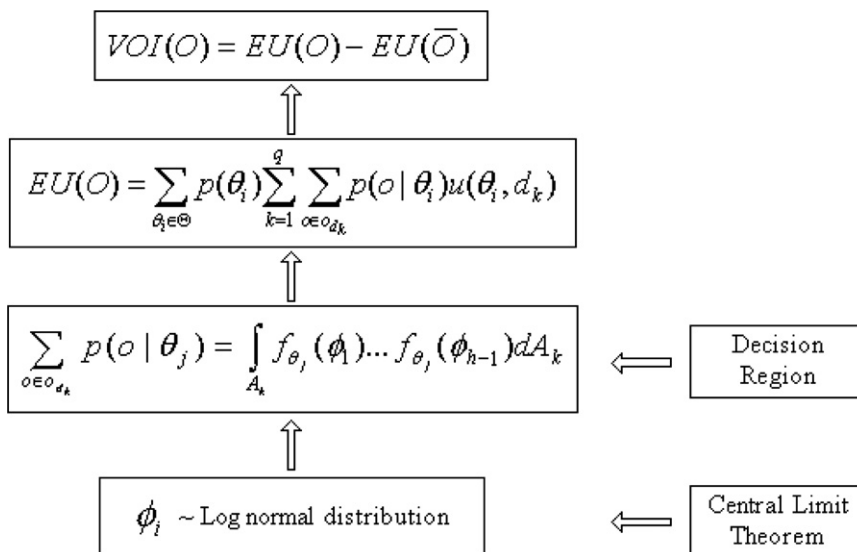


Fig. 8. The key equations to compute VOI when θ has multiple states.

Table 3
ID structures

k	5	4	3	2	1
Number of IDs	2	3	3	1	1

k is the size of the biggest group after partitioning.

Table 4
Testing cases

<i>ID_indep: 2-state</i>	50 test cases, where O_i nodes are conditionally independent given θ whose state is binary
<i>ID_indep: 3-state</i>	50 test cases, where O_i nodes are conditionally independent given θ who has three states
<i>ID_indep: 4-state</i>	50 test cases, where O_i nodes are conditionally independent given θ who has four states
<i>ID_dep: 2-state</i>	450 test cases, where O_i nodes are conditionally dependent given θ whose state is binary
<i>ID_dep: 3-state</i>	450 test cases, where O_i nodes are conditionally dependent given θ who has three states
<i>ID_dep: 4-state</i>	450 test cases, where O_i nodes are conditionally dependent given θ who has four states

4. Experiments

The experiments are designed to demonstrate the performance of the proposed algorithm compared to the exact VOI computation. We limit the ID test model with at most 5 layers² and up to 11 information sources due to the exponential computational time behind the exact computation. Ten different ID models are constructed, where in one of the IDs the O nodes are conditionally independent given the θ node. Table 3 describes the structures of these IDs. The IDs are parameterized with 150 sets of different conditional probability tables and utility functions, a process which yields 1500 test cases. In each the one-third of them, θ node has 2, 3, and 4 states, respectively. Without loss of generality, all the other random nodes and the decision node have four states.

For each test case, the VOIs for different O subsets with the size from 3 to 11 are computed. The results from the approximation algorithm are compared to the exact computation implemented with the brute-forth method. Let VOI_t be the ground-truth, and VOI be the value computed with the proposed algorithm. Assuming $VOI_t \neq 0$, the error rate is defined as follows:

$$Err = \frac{|VOI_t - VOI|}{VOI_t}.$$

The 1500 test cases described previously are divided into six groups, named as *ID_indep: 2-state*, *ID_indep:3-state*, *ID_indep:4-state*, *ID_dep:2-state*, *ID_dep: 3-state*, and *ID_dep:4-state*. Table 4 describes the six groups.

Fig. 9 illustrates the results from the six groups of 1500 test cases. Chart (a) shows the average errors for each group, while Chart (b) shows the VOIs for one specific case, which is randomly chosen from the test cases from *ID_dep: 3-state*. As the set size of the O_i nodes increases, the error rate decreases. When the state number of θ is the same, the error rates of the dependent cases are larger than the error rates of the conditional independent cases. This can be explained by the reason that the IDs in the dependent cases have fewer independent O subsets than the ID in the independent groups. Since the central-limit theorem is the basis of our algorithm, it works better when the number of w_i increases, which corresponds to the number of independent O subsets. Even when the size of O set is as small as 6, the average error is less than or around 0.1 for all the cases. We could run several larger IDs with much more O_i nodes, and the error curve would be progressively decreasing. Here, we intend to show the trend and the capability of this algorithm.

Charts (c) and (d) show the average computational time with the exact computation and the approximation computation. When the set size of the O_i nodes is small, the computational time is similar. However, as the size becomes larger, the computational time of the exact computation increases exponentially, while the computational time of the approximation algorithm increases much slower. Thus, the larger the O set size is, the more time the approximation algorithm can save. Likewise, as the number of the state of each O_i node further increases, the computational saving would be more significant. As the number of states of θ increase, the computational time also slightly increases.

5. An illustrative application

We use a real-world application in human computer interaction to demonstrate the advantages of the proposed algorithm. Fig. 10 shows an ID for user stress recognition and user assistance. The diagram consists of two portions. The upper portion, from the top to the “stress” node, depicts the elements that can alter human stress. These elements include the workload, the environmental context, specific character of the user such as his/her trait, and importance of the goal that

² The length of the longest path starting from (or ending at) the hypothesis node is 5.

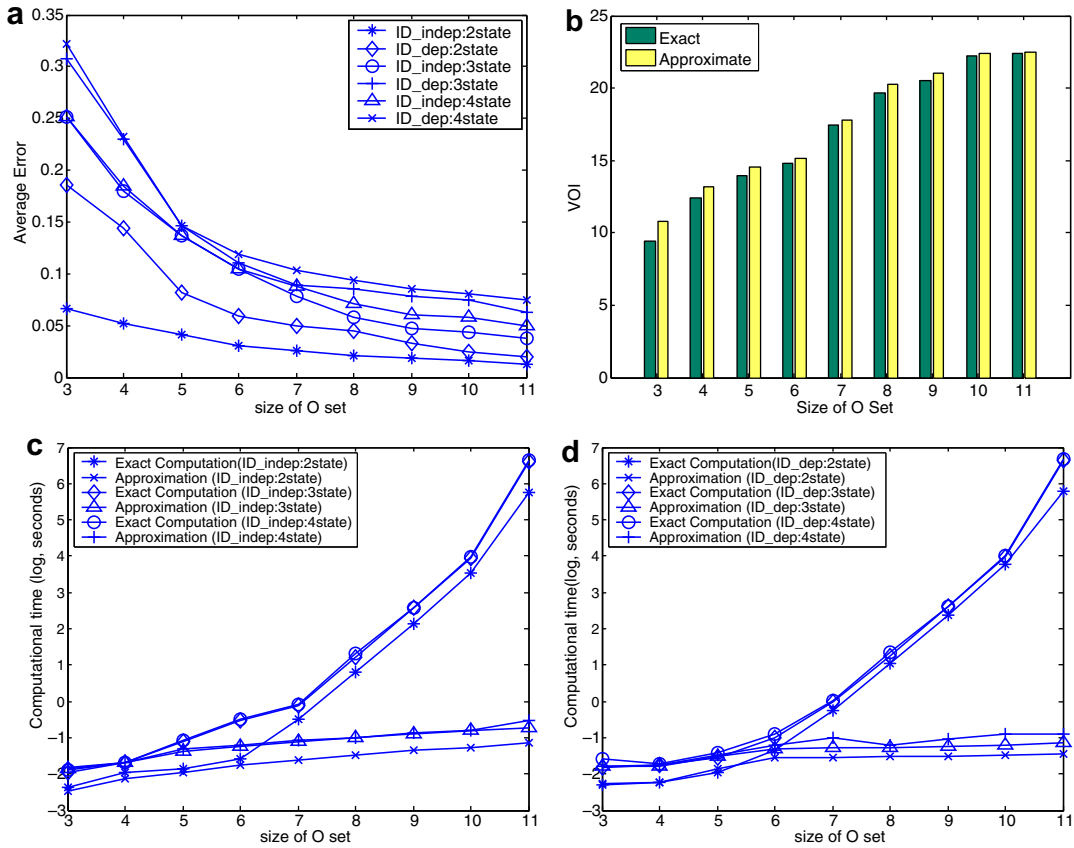


Fig. 9. Results from the four groups of 1500 test cases: (a) average error rates with the approximation algorithm; (b) VOI vs. VOI for one test case from *ID_dep: 3-state*; (c) computational time (log(*t*), unit is second) for the groups of *ID_indep:n-state*, *n* = 2, 3, 4; and (d) computational time (log(*t*), unit is second) for the groups of *ID_dep:n-state*, *n* = 2, 3, 4.

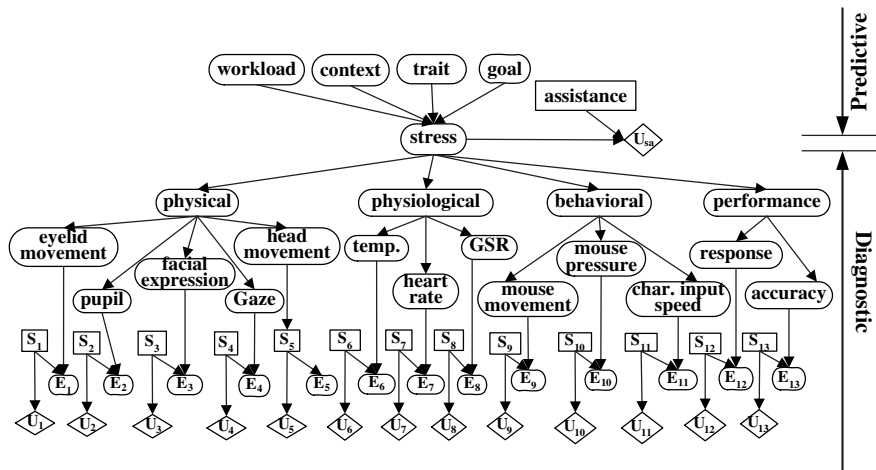


Fig. 10. An influence diagram for recognizing human stress and providing user assistance. Ellipses denote chance nodes, rectangles denote decision nodes, and diamonds denote utility nodes. All the chance nodes have three states.

he/she is pursuing. This portion is called predictive portion. On the other hand, the lower portion of the diagram, from the “stress” node to the leaf nodes, depicts the observable features that reveal stress. These features include the quantifiable measures on the user physical appearance, physiology, behaviors, and performance. This portion is called diagnostic portion.

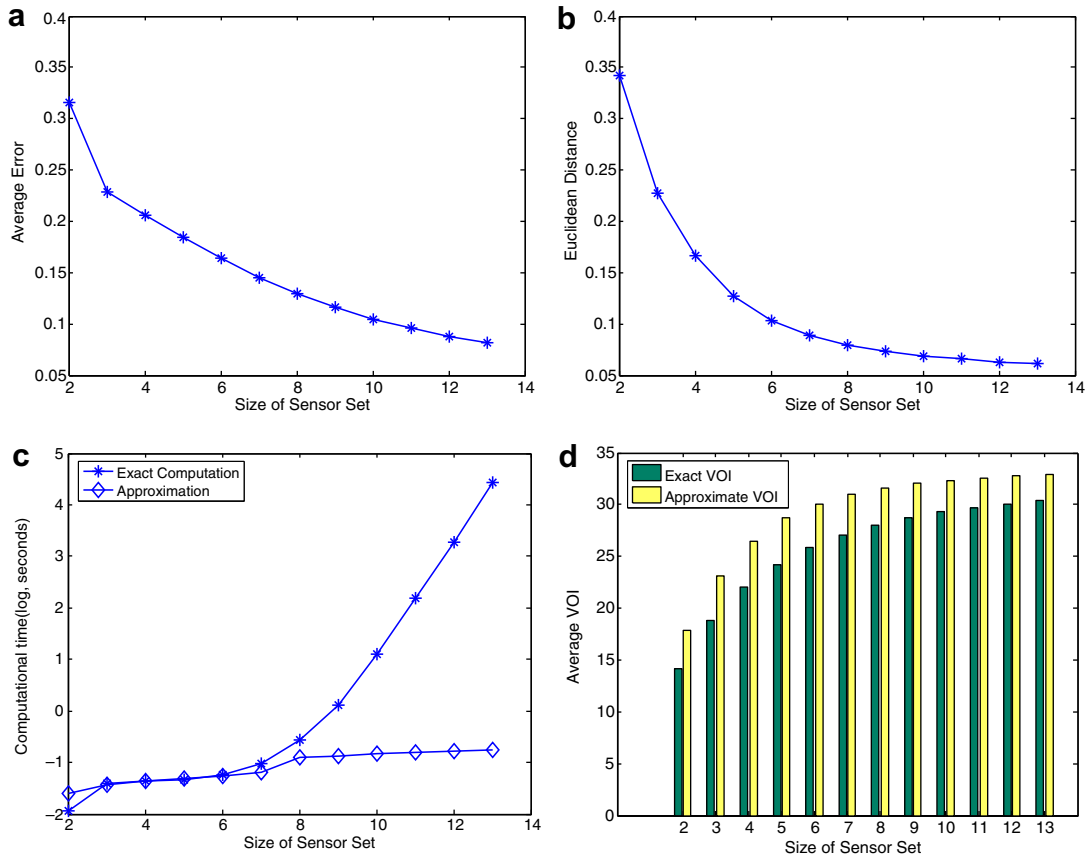


Fig. 11. Results for the stress modeling: (a) average errors with the approximation algorithm; (b) Euclidean distance between the true and approximated $\sum_{o \in o_{d_k}} p(o|\theta_i)$; (c) computational time ($\log(t)$, unit is second); (d) true VOI vs. approximated VOI.

The hybrid structure enables the ID to combine the predictive factors and observable evidence in user stress inference. For more detail please refer to [19].

To provide timely and appropriate assistance to relieve stress, two types of decision nodes are embedded in the model to achieve this goal. The first type is the assistance node associated with the stress node, which includes three types of assistance that have different degrees of impact and intrusiveness to a user. Another type of decision nodes is the sensing action node (S_i node in Fig. 10). It decides whether to activate a sensor for collecting evidence or not. Through the ID, we decide the sensing actions and the assistance action sequentially. In order to first determine the sensing actions (which sensors should be turned on), VOI is computed for a set S consisting of S_i . Using the notations defined before, we have $VOI(S) = VOI(E) - \sum_{S_i \in S} u_i(S_i)$, where E is the set of observations corresponding to S and $VOI(E) = EU(E) - EU(\bar{E})$.

Fig. 11 shows the experimental results for the stress model. We enumerate all the possible combinations of sensors and then compute the value-of-information for each combination. Chart (a) illustrates the average VOI errors for different sensor sets with the same size. And Chart (b) displays the Euclidean distance between the true and estimated probabilities $\sum_{o \in o_{d_k}} p(o|\theta_i)$ (Eq. (26)). Similarly to the simulation experiments, the error decreases as the size of O set increases, and the computational time increases almost linearly in the approximation algorithm.

6. Conclusions and future work

As a concept commonly used in influence diagrams, VOI is widely used as a criterion to rate the usefulness of various information sources, and to decide whether pieces of evidence are worth acquiring before actually using the information sources. Due to the exponential time complexity of computing non-myopic VOI for multiple information sources, most researchers focus on the myopic VOI, which requires the assumptions (“No competition” and “One-step horizon”) that may not meet the requirements of real-world applications.

We thus proposed an algorithm to approximately compute non-myopic VOI efficiently by utilizing the central-limit theorem. Although it is motivated by the method of [11], it overcomes the limitations of their method, and works for more general cases, specifically, no binary-state assumption for all the nodes and no conditional-independence assumption for the

Table 5

The proposed algorithm vs. the algorithm in [11]

Our algorithm	Heckerman's algorithm
Hypothesis node (Θ) can be multiple states	Θ has to be binary
Decision node (D) can have multiple rules	D has to be binary
Information sources nodes (O s) can be dependent from each other	O s have to be conditionally independent from each other

information sources. Table 5 compares our method with the method in [11]. Due to the benefits of our method, it can be applied to a much broader field. The experiments demonstrate that the proposed algorithm can approximate the true non-myopic VOI well, even with a small number of observations. The efficiency of the algorithm makes it a feasible solution in various applications when efficiently evaluating a lot of information sources is necessary.

Nevertheless, the proposed algorithm focuses on the influence diagrams with one decision node under certain assumptions. For example, currently, we assume the hypothesis node Θ and the decision node d are independent. If D and Θ are dependent, but conditionally independent given the observation set O , Eqs. (5) and (6) will not be affected, so our algorithm can still apply. However, if D and Θ are dependent given O , it may be difficult to directly apply our algorithm. Another scenario is that when there are more than one hypothesis node and/or utility nodes. One possible solution is to group all these hypotheses nodes into one. We would like to study these issues in the future.

Appendix

Proposition 1. Let $r_{jk} = \frac{u_{2j} - u_{2k}}{u_{1k} - u_{1j} + u_{2j} - u_{2k}}$, $p_{kl}^* = \max_{k < j \leq q} r_{jk}$, and $p_{ku}^* = \min_{1 \leq j < k} r_{jk}$, then d_k is the optimal action if and only if $p_{kl}^* \leq p(\theta_1) \leq p_{ku}^*$.

Proof of Proposition 1. \Rightarrow In this direction, we prove that if d_k is the optimal action, $p(\theta_1) \geq \max_{k < j \leq q} r_{jk}$ and $p(\theta_1) \leq \min_{1 \leq j < k} r_{jk}$.

If d_k is the optimal action, $EU(d_k)$ must be larger than or equal to the expected utility of any other action. Based on the definition, the expected utility of taking the action d_k is $EU(d_k) = p(\theta_1) * u_{1k} + p(\theta_2) * u_{2k}$, where $u_{1k} = u(\theta_1, d_k)$, and $u_{2k} = u(\theta_2, d_k)$. Therefore, we get the equations as follows:

$$EU(d_k) \geq EU(d_j) \quad \forall j, j \neq k, \tag{27}$$

$$\Rightarrow p(\theta_1) * u_{1k} + p(\theta_2) * u_{2k} \geq p(\theta_1) * u_{1j} + p(\theta_2) * u_{2j}, \tag{28}$$

$$\Rightarrow p(\theta_1) \geq \frac{u_{2j} - u_{2k}}{u_{1k} - u_{1j} + u_{2j} - u_{2k}} = r_{jk} \quad \text{if } j > k, \tag{29}$$

$$p(\theta_1) \leq \frac{u_{2j} - u_{2k}}{u_{1k} - u_{1j} + u_{2j} - u_{2k}} = r_{jk} \quad \text{if } j < k. \tag{30}$$

Thus, based on the above equations, $p(\theta_1) \geq \max_{k < j \leq q} r_{jk}$ and $p(\theta_1) \leq \min_{1 \leq j < k} r_{jk}$.

\Leftarrow In this direction, we prove that if $p(\theta_1) \geq \max_{k < j \leq q} r_{jk}$ and $p(\theta_1) \leq \min_{1 \leq j < k} r_{jk}$, then d_k is the optimal action.

If $p(\theta_1) \geq \max_{k < j \leq q} r_{jk} \quad \forall j, k < j \leq q$, we get

$$p(\theta_1) \geq r_{jk} = \frac{u_{2j} - u_{2k}}{u_{1k} - u_{1j} + u_{2j} - u_{2k}}, \tag{31}$$

$$\Rightarrow p(\theta_1)(u_{1k} - u_{1j} + u_{2j} - u_{2k}) \geq u_{2j} - u_{2k}, \tag{32}$$

$$\Rightarrow p(\theta_1) * u_{1k} + (1 - p(\theta_1)) * u_{2k} \geq p(\theta_1) * u_{1j} + (1 - p(\theta_1)) * u_{2j}, \tag{33}$$

$$\Rightarrow EU(d_k) \geq EU(d_j). \tag{34}$$

Similarly, for $\forall j, 1 \leq j < k$, we can get $EU(d_k) \geq EU(d_j)$. Therefore, d_k has the maximal expected utility and thus is the optimal decision.

Proposition 2. $\sum_{o \in o_{d_k}} p(o) = p(p_{kl}^* \leq p(\theta_1|o) \leq p_{ku}^*)$.

Proof of Proposition 2. Based on Proposition 1, d_k is the optimal decision if and only if the value of $p(\theta_1)$ is between p_{kl}^* and p_{ku}^* . Therefore, given an instantiation o , the probability that d_k is the optimal decision is equal to the probability that $p(\theta_1|o)$ is between p_{kl}^* and p_{ku}^* , i.e., $p(p_{kl}^* \leq p(\theta_1|o) \leq p_{ku}^*)$.

On the other hand, we know that o_{d_k} is a subset of instantiations, each of which corresponds to the optimal action d_k . Therefore, as long as o belongs to the set of o_{d_k} , d_k will be the optimal decision. In other words, the probability of d_k being the optimal decision is the sum of the probability of each $o \in o_{d_k}$, which is $\sum_{o \in o_{d_k}} p(o)$. Therefore, $\sum_{o \in o_{d_k}} p(o) = p(p_{kl}^* \leq p(\theta_1|o) \leq p_{ku}^*)$.

References

- [1] N. Balakrishnan, W.W.S. Chen, *Handbook of Tables for Order Statistics from Log Normal Distributions with Applications*, Kluwer, Amsterdam, Netherlands, 1999.
- [2] G. Casella, R. Berger, *Statistical Inference*, Brooks/Cole, 1990. pp. 45–46 (Chapter 2).
- [3] E. Cohen, N. Megiddo, Improved algorithms for linear inequalities with two variables per inequality, *SIAM Journal on Computing* 23 (6) (1994) 1313–1347.
- [4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 2002.
- [5] E. Crow, K. Shimizu, *Lognormal Distributions: Theory and Applications*, Dekker, New York, 1988.
- [6] M. Diehl, Y. Haimes, Influence diagrams with multiple objectives and tradeoff analysis, *IEEE Transactions on Systems, Man and Cybernetics, Part A* 34 (3) (2004) 293–304.
- [7] S. Dittmer, F. Jensen, Myopic value of information in influence diagrams, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* (1997) 142–149.
- [8] K.J. Ezawa, Evidence propagation and value of evidence on influence diagrams, *Operations Research* 46 (1) (1998) 73–83.
- [9] W. Feller, *An Introduction to Probability Theory and Its Applications*, third ed., vol. 2, Wiley, New York, 1971.
- [10] J. Hammersley, Monte Carlo methods for solving multivariable problems, *Annals of the New York Academy Sciences* 86 (1960) 844–874.
- [11] D. Heckerman, E. Horvitz, B. Middleton, An approximate nonmyopic computation for value of information, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (3) (1993) 292–298.
- [12] R. Howard, Value of information lotteries, *IEEE Transactions of Systems Science and Cybernetics* 3 (1) (1967) 54–60.
- [13] R. Howard, J. Matheson, Influence diagrams, *Readings on the Principles and Applications of Decision Analysis* 2 (1981) 721–762.
- [14] F. Jensen, *Bayesian Networks and Decision Graphs*, Springer-Verlag, New York, 2001.
- [15] F. Jensen, F.V. Jensen, S. Dittmer, From influence diagrams to junction trees, (1994) 367–374.
- [16] M. Kalos, P. Whitlock, *Monte Carlo Methods*, Wiley, New York, 1986.
- [17] K.B. Korb, A.E. Nicholson, *Bayesian Artificial Intelligence*, Chapman and Hall/CRC, 2003.
- [18] W. Liao, Q. Ji, Efficient active fusion for decision-making via VOI approximation. *Twenty-first National Conference on Artificial Intelligence (AAAI)*, 2006.
- [19] W. Liao, W. Zhang, Z. Zhu, Q. Ji, A decision theoretic model for stress recognition and user assistance, *Twentieth National Conference on Artificial Intelligence (AAAI)* (2005) 529–534.
- [20] Mathematica, 2006. <<http://www.wolfram.com/products/mathematica/index.html>>.
- [21] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann Publishers, 1988.
- [22] K.L. Poh, E. Horvitz, A graph-theoretic analysis of information value, *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)* (1996) 427–435.
- [23] H. Raiffa, *Decision Analysis*, Addison-Wesley, 1968.
- [24] R. Shachter, Evaluating influence diagrams, *Operations Research* 34 (6) (1986) 871–882.
- [25] R. Shachter, Efficient value of information computation, *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)* (1999) 594–601.
- [26] R. Shachter, P.M. Ndilikilikesha, Using potential influence diagrams for probabilistic inference and decision making, *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI-93)* (1993) 383–390.
- [27] F. Yokota, K.M. Thompson, Value of information analysis in environmental health risk management decisions: past, present, and future, *Risk Analysis* 24 (2004) 635–650.
- [28] N.L. Zhang, R. Qi, D. Poole, Incremental computation of the value of perfect information in stepwise-decomposable influence diagrams, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence* (1993) 400–410.

Active and Dynamic Information Fusion for Multisensor Systems With Dynamic Bayesian Networks

Yongmian Zhang and Qiang Ji

Abstract—Many information fusion applications are often characterized by a high degree of complexity because: 1) data are often acquired from sensors of different modalities and with different degrees of uncertainty; 2) decisions must be made efficiently; and 3) the world situation evolves over time. To address these issues, we propose an information fusion framework based on dynamic Bayesian networks to provide active, dynamic, purposive and sufficing information fusion in order to arrive at a reliable conclusion with reasonable time and limited resources. The proposed framework is suited to applications where the decision must be made efficiently from dynamically available information of diverse and disparate sources.

Index Terms—Active sensing, Bayesian networks, information fusion.

I. INTRODUCTION

There has been a great deal of interest in the development of systems capable of using many different sources of sensory information [1], [2]. Whatever the application may be, there is a need to systematically and efficiently interpret the large volume of information acquired from sensors of different modalities and with different degrees of uncertainty. Typically, many applications contain a large number of uncertain events interrelated by causes and effects. The question is how to systematically and efficiently represent and fuse uncertain information at different levels of abstraction.

The world situation is often dynamic and uncertain in nature and unfolds over time. To correctly assess and interpret the dynamic environment, a fusion system is needed that not only can systematically handle uncertain sensory data of different modalities but, more importantly, can reason over time. The inability of current sensor fusion systems to correlate and reason about a vast amount of information over time is an impediment to providing a coherent overview of the unfolding events. The question is, therefore, how to account for the temporal changes in sensory information.

Moreover, many applications are often constrained by limited time and resources. The usage of more sensors incurs more cost in acquiring information. It is important to avoid unnecessary or unproductive sensor actions and computations. Thus, we must select a subset of sensors that are the most decision-relevant. Now the question is how to determine a set of most informative information sources for the current goal with minimal cost at particular stage of information gathering to achieve an efficient and timely decision.

To address above issues, a fusion system therefore requires the capability which can not only represent the temporal changes of uncertain sensory information, but dynamically select the most relevant sensory data for a given goal at a given time as well. To achieve this, we propose to cast information fusion into a framework of dynamic Bayesian networks (DBNs) to account for the temporal aspect of decision making and uncertainty of knowledge, and to integrate and infer dynamic sensory information of different modalities. The fusion system is able to

actively select a subset of sensors to produce the most decision-relevant information with limited resources and in reasonable time. Additionally, DBNs enable sensor selection to dynamically adapt to varying situations.

Research in information fusion is getting wider and deeper nowadays. There are a number of methods available for sensor fusion including evidential reasoning [3], fuzzy theories [4]–[6], and neural networks [7], [8]. However, these methods lack the sufficient expressive power to handle uncertainties, dependencies and dynamics exhibited by sensory data in many applications. Bayesian networks (BNs), since their inception, have shown great promise in performing multisensor data fusion [1]. Recently, DBNs extend BNs for modeling dynamic events [9]–[13]. In the existing works, BNs and DBNs are primarily used for knowledge and uncertainty representation. DBNs were proposed as a generalization of hidden Markov models (HMMs) [14], but they allow much more general graph structures than an HMM does. It is therefore natural to consider a DBN as a basis of the general spatio-temporal sensor data analysis and interpretation.

Active sensing involves actively controlling sensors to optimize information gathering in a knowledge-based manner with an identifiable selection criterion rather than randomly selecting sensor parameters. A significant amount of research has been directed to the area of computer vision and robotics [15]–[19]. More recently, Oliver *et al.* [20] studied selective perception policies to purposively guide sensing and visual information processing to circumvent the computational burden associated with perceptual analysis. Pinz *et al.* [21] is the first to introduce active fusion in remote sensing image understanding. Paletta and Pinz proposed an active recognition system [22], where mutual information is used to quantify the ambiguity and to determine which view to select. Denzler and Brown [23] applied mutual information theory in the state estimation process for active camera parameter selection. Using HMMs and evidential reasoning to combine instantaneous and temporal visual information is recently studied in [24]. The notion of dynamically combining information provides useful hint to this work. A central problem of active information fusion is the sensor selection problem. There have been several attempts in sensor selection for target localization [25]–[28]. In these works, they assumed that the system model is either a linear system or a standard stochastic model. Other works of interest for sensor selection can be found in the control literature. The work presented in [29] considers an Hidden Markov model with a number of sensors. The sensor sequence is determined via stochastic dynamic programming. The problem of optimal sensor selection for discrete-event systems under partial observation was studied in [30]. Since we use a DBN as an information fusion method, the above approaches of sensor selection can not directly be applied to our problem.

This paper aims to formalize a framework based on DBNs for active and dynamic information fusion, which is particularly suited to the applications where the decision must be made efficiently from dynamically available information of diverse and disparate sources. The remainder of this paper is organized as follows. We start with information fusion using DBNs in the next section. An active and dynamic information fusion framework is proposed in Section III. Section IV presents an example for a proof-of-concept. The final section is the conclusion.

II. INFORMATION FUSION WITH BAYESIAN NETWORKS

Information fusion is a process dealing with the association, correlation, and combination of information collected from various disparate sources into one coherent structure that can be used by a computer system to make a better decision than from single source only. In order

Manuscript received December 21, 2004; revised April 17, 2005. This work was supported by a Grant from the U.S. Army Research Office under Grant DAAD19-01-1-0402. This paper was recommended by Associate Editor Isabelle Bloch.

The authors are with the Department of Electrical, Computer and System Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: jiq@rpi.edu).

Digital Object Identifier 10.1109/TSMCB.2005.859081

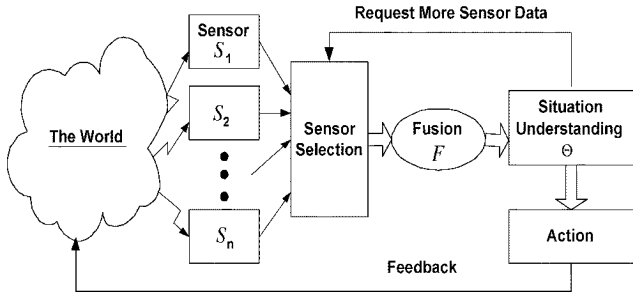


Fig. 1. Overview of a dynamic information fusion process in sequential decision-making for situation understanding. The figure shows that, if the certainty about the world situation is not sufficiently high, the additional information is requested by selecting a set of information sources. When knowing enough about the world situation, the decision-maker may make a decision about which course of action corresponding to that type of situation needs to be taken.

to accomplish this task, a fusion function F must be used to combine these multiple sensor readings to a single output. Regardless of the fusion structure, the fusion process can be generally expressed as

$$\Theta = F(S_1, S_2, \dots, S_n) \quad (1)$$

where S_i denotes an individual sensor that gives a measurement; Θ is the output on which the decision-making is based. The choice of the fusion function F depends on the chosen fusion methods. A review of information combination methods can be found in [31]. The fusion method in this effort is a dynamic probabilistic network; while the probabilistic inference is analogous to the fusion function F . The output Θ is the posterior probability of hypotheses that we want to infer. Fig. 1 outlines an active fusion process in sequential decision-making for situation understanding.

In terms of probabilistic networks, chain graphs have been explored to incorporate probabilistic reasoning for data analysis tools [32]–[34]. A chain graph is a probabilistic network model that mixes undirected and directed graphs to give a probabilistic representation. However, in order to represent the causality between random variables as well as time-series data, it is natural to use directed graphical models, which can capture the fact that one causes another as well as time flows forward. BNs or DBNs are ideal for representing directed graphs.

A Bayesian network is a graphical model representing probabilistic relationships among a set of variables to reflect an expert's understanding of the domain. A rigorous definition of Bayesian networks can be found in [35]. Here we develop the concept with just enough rigor and detail that will enable us to apply them to information fusion problems. A general definition of a BN, as shown in Fig. 2(a), is given as follows. Let (\mathcal{E}, P) be a joint probability space with $\mathcal{E} = \mathcal{E}_1 \times \dots \times \mathcal{E}_n$, and joint probability P . Given a directed acyclic graph (DAG) of $G = (X, E)$ with $X = \{X_1, \dots, X_n\}$ and arcs E , where X_i is the projection onto \mathcal{E}_i . Let $\pi(X_i)$ be the parents of X_i and $A(X_i)$ be the nondescendant of $\pi(X_i)$. Then (G, P) is a Bayesian network if, for all $X_i \in X$, X_i and $A(X_i)$ are independent given $\pi(X_i)$. The resulting joint probability over the random variables in the network can be expressed as

$$P(X) = \prod_{i=1}^n P(X_i | \pi(X_i)). \quad (2)$$

Bayesian networks are valuable for several reasons. First, they enable to model the dependencies and uncertainties of the events and to handle incomplete data sets without difficulty because they discover dependencies among all variables. Second, domain knowledge can be described in a hierarchical graphical structure to represent different levels

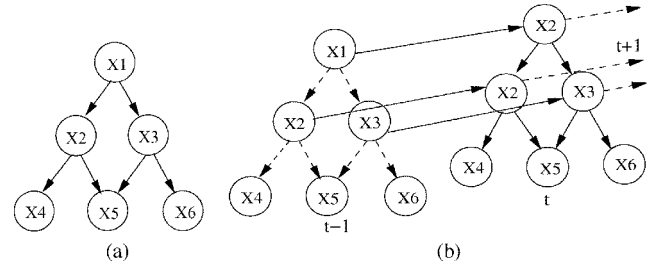


Fig. 2. (a) Static Bayesian network (a directed acyclic graph), where $X_1 - X_6$ are random variables. (b) A DBN is defined by "unrolling" the two-slice BN. Assume that the model is first-order Markov. The figure shows that X_1 , X_2 and X_3 at current time t are connected to their corresponding variables which determine the system at previous moment of time $t - 1$.

of abstraction. Third, they provide a mathematically rigorous foundation for consistent, coherent and efficient reasoning.

BNs were initially not suited to modeling dynamic events. To circumvent this limitation, a new statistical approach from the perspective of BNs was proposed as a generalization of Kalman filtering models (KFMs) [36] and HMMs [37], namely, DBNs [14]. A DBN model is made up of interconnected two time slices of a static BN, and the transition of BN between the two consecutive time t and $t + 1$ satisfies the Markov process. Conventionally, it is assumed that a DBN is first-order Markov, and that temporal nodes¹ at the current time t are connected only to the corresponding nodes at the next time slice $t + 1$. DBNs, however, can be extended by connecting a node at t to any nodes at $t + 1$ it may affect. However, such connections not only greatly complicate the network topology but also have limited utility since the impact of a node at t on any nodes other than itself at $t + 1$ are usually accounted for through the propagation of its influence on the corresponding node to the other nodes. Therefore, DBNs can be implemented by keeping in memory two slices at any one time, representing previous time slice and current time slice, respectively. The nodes in the first time slice do not have any parameters associated with them and they only determine the system at the previous moment of time; while each node from the second time slice has an associated conditional probability distribution. The two slices are such rotated that old slices are dropped and new slices are used as time progresses. The arcs between slices are from left to right, reflecting the temporal causality and they are parameterized by transitional probabilities. We only consider discrete-time stochastic processes, so we increase the index t by one every time a new observation arrives. Fig. 2(b) shows an example of DBNs as given by the definition above. The joint distribution from the initial moment of time ($t = 1$) until the time boundary ($t = T$) is then given by

$$P(X_{1:T}) = \prod_{t=1}^T \prod_{i=1}^n P(X_i^t | \pi(X_i^t)) \quad (3)$$

where X_i^t is the i 'th node at time t ; $\pi(X_i^t)$ stands for the parents of a node X_i at time t , and they can either be in the same time slice or in the previous time slice. The difference between a DBN and an HMM is that a DBN represents the hidden state in terms of a set of random variables. By contrast, in an HMM, the state space consists of a single random variable. The difference between a DBN and a KFM is that a KFM requires all the conditional probability densities (CPDs) to be linear-Gaussian, whereas a DBN allows general hybrid, nonlinear CPDs. In addition, HMMs and KFMs have a restricted topology, whereas a DBN allows much more general graph structures. Therefore, it is natural to consider a DBN as a basis of the general spatio-temporal sensor data

¹Temporal nodes represent variables that evolve over time.

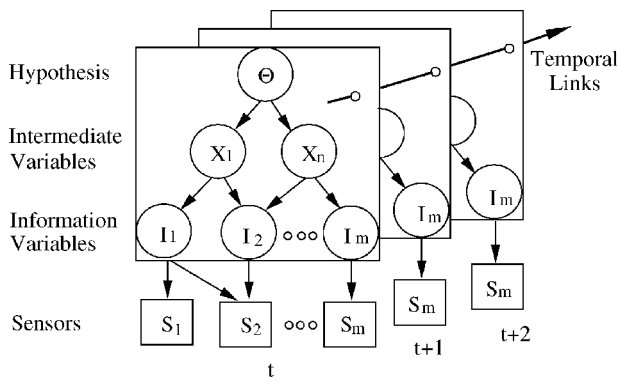


Fig. 3. Sensors are viewed as random variables and incorporated into a Bayesian network to form a coherent information fusion structure. For representing sensor uncertainties, a layer of information variables $\{I_1, \dots, I_m\}$ is added into dynamic Bayesian networks to interface sensors and intermediate variables.

analysis and interpretation. A thorough work on DBN representation, inference and learning can be found in [13].

With the hypothesis and sensors, we can construct a coherent fusion structure with a Bayesian network as shown in Fig. 3. The root node of such a network would contain the hypothesis variable whose states corresponds to multiple hypotheses. The sensors occupy the lowest level nodes without any children. Sensors are the only observable variables in the model and evidences are gathered through sensors. In general a network will have a number of intermediate nodes that are interrelated by cause and effect. The hypothesis node is causally linked to the sensor nodes through these intermediate variables. The nodes and the links should reflect the causal structure and context independencies pertaining to the problem we are modeling.

In the real world, however, a fusion system may receive incorrect information from sensors for various reasons such as sensor noise, imprecise acquisition devices, and limitations of numerical reconstruction algorithms. If information is expert knowledge, experts also differ in their level of expertise. Therefore, sensor readings include uncertainties, which may diminish the reliability of a fusion system. There are a number of ways to represent uncertainty in sensory data [38]. Nevertheless, a probabilistic metric provides us with some consistency for information throughout the probabilistic graphical model. The uncertainty of sensor readings measures the degree of belief that the information provided by a sensor reflects the actual value. To be able to handle the uncertainty of sensor readings in the fusion structure with a probabilistic network, we may add an additional layer of variables which connects sensors to intermediate variables, namely information variables, as shown in Fig. 3. Conditional probabilities between information variables and sensors quantify the uncertainty of sensor measurements. Consequently, the uncertainty of sensor readings is incorporated into the fusion system to update the probability distribution over the hypothesis variable. Evidences regarding information variables are gathered through sensors and are fused through DBN inference. Temporal link between two consecutive time slices reflects the temporal causality. The time t increases by one every time new sensor information arrives.

III. ACTIVE AND DYNAMIC INFORMATION FUSION

The objective of active information fusion is to selectively choose the most decision-relevant information while minimizing the cost associated with using the sensors for acquiring information. Overall efficiency can be achieved by aggregating only a subset of the most relevant sensor data to address the current goal. In other words, active fusion focuses on what is optimal rather than what is available.

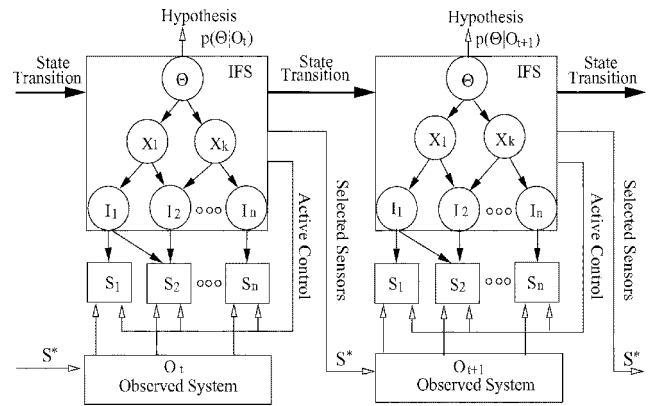


Fig. 4. Functional view of active information fusion, in which the fusion structure is a Bayesian network consisting of hypotheses Θ , intermediate variables X , information variables I , and sensors S . The closed arrows represent the causality relations and the open arrows represent sensor activation control.

The problem of active fusion can be stated mathematically as follows. Assume that there are m sensors $S_i, i = 1, \dots, m$ available that can be used to give measurements of the environment. Let Θ be a set of hypothesis $\theta_k, k = 1, \dots, K$. Let the sensors $\mathbf{S} = \{S_1, \dots, S_n\}$ be a subset of sensors selected at time t , where $n \in \{1, \dots, m\}$. The measurement of a sensor S_i at time t is denoted as $o_t(S_i)$, and $o_t(S_i)$ of the i 'th sensor belongs to a known finite set of states $e_1^{(i)}, \dots, e_L^{(i)}$. That is, the i 'th sensor can yield one of L possible measurements at a given time instant t . Let $O_t = \{o_t(S_1), \dots, o_t(S_n)\}$ represent the information available at current time t upon which the sensor selection is based at time $t + 1$. The active information fusion generally proceeds in four stages at each time instant.

- (1) *Sensor Selection*: based on the system state after receiving O_t , select an optimal subset of sensors \mathbf{S}^* to be activated at the next time step $t + 1$.
- (2) *Observation*: get observation $o_{t+1}(S_i), i = 1, \dots, n$, to obtain new sensory information O_{t+1} , where $S_i \in \mathbf{S}^*$.
- (3) *State Estimation*: compute the posterior probability $p(\Theta_{t+1}|O_{t+1})$ by using DBN inference algorithms.
- (4) *Decision-making*: make a decision if the certainty in the current solution is sufficiently high. Otherwise, start over and select sensors for further observations.

To determine a set of sensors to activate, it can only consider the probable outcomes of sensors. The actual outcomes of sensors can only be determined once the sensors are instantiated. Fig. 4 provides an architectural concept for the framework of active and dynamic information fusion. On the basis of this figure, the active control is to select a subset of sensors to be activated for the next time instant by only considering the probable outcomes of sensors (not physically invoked). The selected sensors have the greatest expected contribution to the uncertainty reduction (compared to their costs). The observed system is to physically invoke the selected sensors at time t , and generates sensory information O_t , the set of actual outcomes of selected sensors.

Acquiring information incurs cost such as operational cost, computational cost, etc. In a military context, the cost also includes the risk involved in information gathering. It is apparent that any quantitative analysis of information gain must account for the conflicting objective of sensor activation. Therefore, optimal sensor selection is to maximize a utility. In general, the utility function consists of two components: information gain u_1 and the cost $C(\mathbf{S})$ to activate sensors \mathbf{S} . We use $u_2 = 1 - C(\mathbf{S})$ to convert the cost to the cost saving, which makes u_1 and u_2 in qualitative equivalence (both represents the benefit). Since

u_1 and u_2 are mutually utility independent [39], we can design a multilinear utility function as

$$U(u_1, u_2) = (k_1 u_1 + 1)(k_2 u_2 + 1) \quad (4)$$

where k_1 and k_2 are the preference parameters and $k_1 + k_2 = 1$. u_1 and u_2 need to be normalized in quantitative equivalence.

From the viewpoint of information theory, mutual information indeed measures the gain of an average amount of information before and after instantiating the selected sensors. Follow the notations in Fig. 4 and consider the process at certain time instant t . Notice that t is dropped for notational clarity in the following equations. To obtain the mutual information, we shall start with the expected entropy of information. The expected entropy of hypothesis Θ with respect to all possible outcomes of a sensor S_i measures how much uncertainty exists in Θ given S_i , is

$$H(\Theta|S_i) = - \sum_{s_i} \sum_{\Theta} P(\theta, s_i) \log P(\theta|s_i) \quad (5)$$

where s_i denotes the value taken by sensor S_i . If subtracting $H(\Theta|S_i)$ from the original uncertainty in Θ prior to instantiating S_i , $H(\Theta)$, yield the amount of information about Θ that S_i is capable of providing

$$\begin{aligned} I(\Theta; S_i) &= H(\Theta) - H(\Theta|S_i) \\ &= - \sum_{\Theta} P(\theta) \log P(\theta) \\ &\quad + \sum_{s_i} \left\{ P(s_i) \sum_{\Theta} P(\theta|s_i) \log P(\theta|s_i) \right\} \\ &= \sum_{\Theta} \sum_{s_i} P(\theta, s_i) \log \frac{P(\theta|s_i)}{P(\theta)} \end{aligned} \quad (6)$$

where $I(\Theta; S_i)$ also quantifies the total uncertainty-reducing potential of S_i regarding Θ . Then mutual information $I(\Theta; \mathbf{S})$ for a sensor set $\mathbf{S} = \{S_1, \dots, S_n\}$ may be written as

$$\begin{aligned} I(\Theta; \mathbf{S}) &= H(\Theta) - H(\Theta|\mathbf{S}) \\ &= \sum_{\Theta} \sum_{s_1 \dots s_n} \left\{ P(\theta, s_1, \dots, s_n) \log \frac{P(\theta|s_1, \dots, s_n)}{P(\theta)} \right\} \end{aligned} \quad (7)$$

where $P(\theta, s_1, \dots, s_n)$ and $P(\theta|s_1, \dots, s_n)$ at time t can be directly obtained through DBN inference algorithms by considering the state of temporal variables at time $t-1$ and current sensor observations at time t .

Equation (7) provides a selection criterion in identifying the uncertainty reduction capability given a sensor set \mathbf{S} . Unfortunately, it is impractical to simply implement this criterion due to two difficulties when n is large: 1) it requires time exponential in the number of summations to compute mutual information exactly and 2) it is infeasible to identify an optimal subset from a large number of information sources. This computational problem is beyond the scope of this present work as we focus on the principle and issues of active and dynamic information fusion. Now let \mathcal{S} be entire space of sensor subsets and \mathbf{S}^* be an optimal sensor subset. We then summarize a sequential decision process with the proposed framework as follows.

SEQUENTIAL-DECISION-PROCESS

- 1 $t \leftarrow 0$
- 2 $\mathbf{S}^* = \arg \max_{\mathbf{S} \in \mathcal{S}} U(I(\Theta; \mathbf{S}), C(\mathbf{S}))$
- 3 **while** $t < T$
- 4 Activate S_i for each $S_i \in \mathbf{S}$ and get O_t

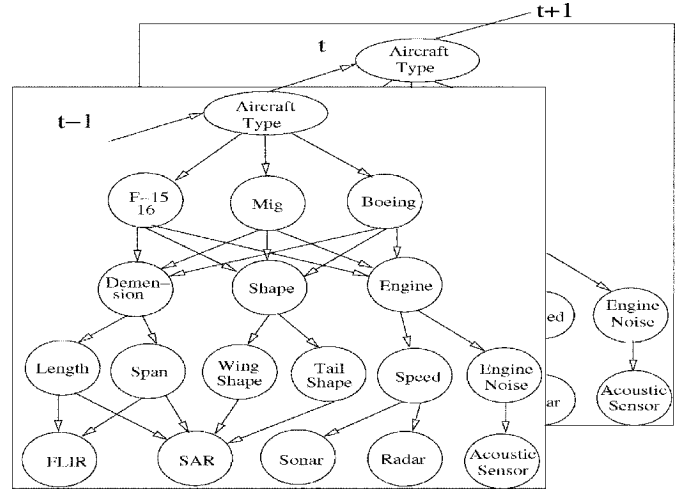


Fig. 5. DBN model for a conceptual IFF system to assist in the identification of aircraft. The transition between two neighboring time slices is modeled by first-order HMM. Assume that only the hypothesis nodes between two time slices are connected.

- 5 Perform DBN inference and obtain $P(\Theta|O_t)$
- 6 **if** confidence is sufficiently high
- 7 make decision
- 8 **else** $\mathbf{S}^* = \arg \max_{\mathbf{S} \in \mathcal{S}} U(I(\Theta; \mathbf{S}), C(\mathbf{S}))$
- 9 $t \leftarrow t + 1$

A strategy of sensor selection needs to be implemented for lines 2 and 8 in the above procedure. Using the brute-force approach or greedy approach for sensor selection, we need to evaluate (7) when information theoretic criterion is used; this is feasible only for the problems with a small number of sensors and with a limited number of sensors being instantiated. To avoid the intractability of exact information computations, myopic is often used under the assumption that the decision maker will act after observing only one sensor. To correctly identify the cost-effective sensors, we should take into account the fact that the decision maker may instantiate more than one sensor before acting. One often used method for selecting multiple sensors at a time is the greedy approach, which greedily selects an ensemble of sensors iteratively until either the combined utility of the selected sensors peaks or when the maximum number of sensors is reached. While efficient, the greedy approach can not guarantee the optimality of the solution. A theoretic approach to the optimal and efficient sensor selection behind this framework is the focus of our current research. Our initial solution to efficiently compute (7) can be seen in [40]. Below, we use a brute-force approach to compute $I(\Theta; \mathbf{S})$ and $U(\cdot)$ with a limited number of sensors being instantiated at each time.

IV. EXAMPLE

To clarify the basic notions, we first experiment with a very simple example for a proof-of-concept. Suppose that a system of identification friend or foe (IFF) employs sensors including imaging sensors (e.g., FLIR, SAR), acoustic sensor, and radar sensor, etc., to identify characteristics of all known aircrafts. Fig. 5 presents a BN model for such a system. The most important identification features that best characterize a particular aircraft include length, span, wing and tail shape, speed, and engine sound, and they are provided by diverse sensors. For examples, to obtain the aircraft shape, we may activate the imaging sensors to determine the shape parameters of the aircraft; while we may

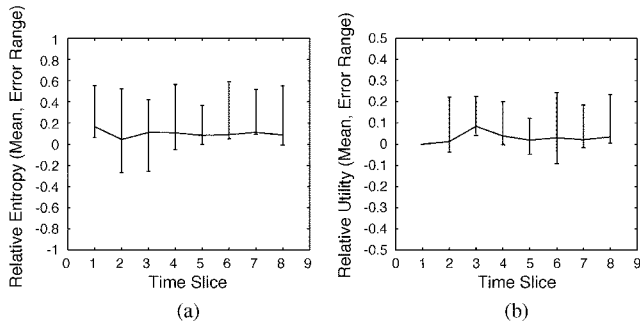


Fig. 6. (a) Comparison of entropy reduction between passive fusion and active fusion. Relative entropy = $(H_2 - H_1)/H_1$, where H_1 and H_2 represent the entropy by active fusion and passive fusion, respectively. The error bar shows the lower and upper error ranges. (b) Comparison of utility between passive fusion and active fusion. The preference parameters k_1 and k_2 take 0.5 in (4). Relative utility = $(U_1 - U_2)/U_2$, where U_1 and U_2 represent the utility by active fusion and passive fusion, respectively. The error bar shows the lower and upper error ranges.

TABLE I
EXAMPLES OF SENSOR ACTION SEQUENCE WHERE SENSOR(S)
ARE ACTIVELY SELECTED AT EACH TIME INSTANT

Time Slice	Example I		Example II	
	Sensor Sequence	Certainty	Sensor Sequence	Certainty
1	Sonar	0.0051	Sonar, FLIR	0.0056
2	FLIR	0.0200	SAR	0.0358
3	Acoustic, Radar	0.2528	Sonar, FLIR	0.0308
4	Acoustic	0.2585	Acoustic, Radar	0.1080
5	FLIR	0.0989	FLIR	0.0723
6	FLIR, Sonar	0.3952	Acoustic, Radar	0.1793
7	SAR	0.1887	Acoustic	0.1571
8	Acoustic, Radar	0.3539	Sonar, FLIR	0.1600

utilize Sonar to obtain the aircraft speed. We assume that there are external modules which receive sensor data and make the data available as input evidence to the network. We shall select the most decision-relevant sensors in order to make a timely engagement decision.

We now specify the cost of individual sensors as [0.1667, 0.2667, 0.2000, 0.1333, 0.2333] for FLIR, SAR, Sonar, Radar, and Acoustic sensor, respectively. Fig. 6(b) shows the comparison result regarding the expected utility between active and passive fusion. For the convenience of comparison, we give the same sensors and sensor outcomes at the first time slice for both active fusion and passive fusion. Fig. 6(b) is the average relative utility of 20 sensor action sequences. As we initially expect, it shows that, on average, the sensors by active selection achieve greater utility than the sensors by passive selection.

Table I gives examples of different courses of sensor actions. In fact, we can see from Table I that the change of situation (here different sensor outcomes) results in different sensor action sequences. At each time instant, the number of sensors and which subset of sensors to be integrated are determined according to the state at that time as well as the evidence at previous time instant. Note that the certainty in Tables I and II is the information gain $I(\Theta; \mathbf{S})$. The larger the information gain, the more certainty there is.

The uncertainty of sensor readings also directly influences the sensor selection. To clarify this matter, we now change the uncertainty of sensor readings (probability distributions between information variables and sensors). Although, at the first time instant, the same sensor and sensor outcome in Table I are given, Table II shows that the course of sensor actions differs with that in Table I due to the change of the uncertainty of sensor readings.

TABLE II
THE SENSOR ACTION SEQUENCE CHANGES DUE TO THE
CHANGE OF SENSOR UNCERTAINTY

Time Slice	Example I		Example II	
	Sensor Sequence	Certainty	Sensor Sequence	Certainty
1	Sonar	0.0056	Sonar, FLIR	0.0471
2	Acoustic	0.0392	Sonar, Acoustic	0.0262
3	Acoustic, Radar	0.0752	Sonar, FLIR	0.0913
4	FLIR, Sonar	0.0448	FLIR	0.0448
5	FLIR	0.0441	FLIR	0.0392
6	FLIR, Sonar	0.0420	Acoustic, Radar	0.1733
7	FLIR	0.1473	Acoustic	0.1529
8	Radar	0.1643	FLIR	0.1571

In a dynamic environment, a decision is required across a fairly narrow space of time, and tasks are dependent on an ongoing, up-to-date analysis of the environment. If the same evidence is acquired sequentially, the evidence acquired at current time can reinforce the hypothesis made by the same information received at previous time. Although a static BN model can integrate all evidences available so far by sequentially propagating the impact of each evidence, the impact of previous evidence on subsequent integration can not be adjusted, and previous evidence can not be integrated into the same evidence currently received. DBNs, on the other hand, enable to correlate and associate the continual arriving evidences through temporal dependencies to perform reasoning over time. The information from previous time serves as prior information for current evidences, and they are combined with Bayesian statistics. Dempster-Shafer evidence theory [3] can also represent the uncertainty of information, but it lacks the ability to handle prior knowledge and temporal dependency.

V. CONCLUSIONS

There are three important issues for a fusion system: 1) modeling uncertainties of sensory data; 2) modeling temporal change; and 3) active control and management of the fusion process. In this paper, we proposed a framework to simultaneously address the above three issues. Toward the first and the second issue, we adopt a dynamic Bayesian network as a fusion structure to provide a coherent and fully unified hierarchical probabilistic framework for representation, integration, and inference of uncertain sensory information of different modalities at different levels of abstraction, and to account for the temporal aspect of decision making. Toward the last issue, we need an efficient sensor selection method that allows a fusion system to actively select and invoke a subset of sensors that is the most decision-relevant.

It is impractical to simply implement information-theoretic criterion in (7) to identify an optimal sensor subset because (7) generally requires time exponential in the number of summations to compute mutual information exactly. In this paper, we mainly focus on the architectural concept and issues for active and dynamic information fusion rather than the efficient sensor selection methodology behind this framework. In many fusion applications, the relationships among the events are mostly unknown and they may vary as the world situation evolves. Therefore, a fusion system needs to incorporate a situation oriented learning mechanism. This is another issue that we will focus on in our future work.

ACKNOWLEDGMENT

The authors gratefully acknowledge constructive comments from the anonymous reviewers that significantly improved the presentation of this paper.

REFERENCES

- [1] E. Waltz and J. Llinas, *Multisensor Data Fusion*. Norwood, MA: Artech House, 1990.
- [2] D. L. Hall and S. A. H. McMullen, *Mathematical Technique in Multisensor Data Fusion*. Norwood, MA: Artech House, 2004.
- [3] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.
- [4] D. Dubois and H. Prade, "A review of fuzzy set aggregation connectives," *Inform. Sci.*, vol. 36, pp. 85–121, 1985.
- [5] R. R. Yager, "On a general class of fuzzy connectives," *Fuzzy Sets Syst.*, vol. 4, pp. 235–242, 1980.
- [6] L. Valet, G. Mauris, P. Bolon, and N. Keskes, "A fuzzy rule-based interactive fusion system for seismic data analysis," *Inform. Fusion*, vol. 4, no. 2, pp. 123–133, 2003.
- [7] G. A. Carpenter and W. D. Ross, "Art-emap: a neural network architecture for object recognition by evidence accumulation," *IEEE Tran. Neural Netw.*, vol. 6, no. 4, pp. 805–818, Jul. 1995.
- [8] O. Parsons and G. A. Carpenter, "Artmap neural networks for information fusion and data mining: map production and target recognition methodologies," *Neural Netw.*, vol. 16, no. 7, pp. 1075–1089, 2003.
- [9] T. Dean, T. Camus, and J. Kirman, "Sequential decision making for active perception," in *DARPA Image Understanding Workshop*, 1990, pp. 889–894.
- [10] G. M. Provan and J. R. Clarke, "Dynamic network construction and updating techniques for the diagnosis of acute abdominal pain," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 3, pp. 299–307, Mar. 1993.
- [11] G. Zweig, "Speech Recognition With Dynamic Bayesian Networks," Ph.D. dissertation, Univ. of California, Berkeley, 1998.
- [12] V. Pavolovic, "Dynamic Bayesian Networks for Information Fusion With Application to Human-Computer Interfaces," Ph.D. dissertation, Univ. of Illinois at Urbana-Champaign, 1999.
- [13] K. P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," Ph.D. dissertation, Univ. of California, Berkeley, 2002.
- [14] T. Dean and K. Kanazawa, "A model for reasoning about persistence and causation," *Artif. Intell.*, vol. 93, no. 1–2, pp. 1–27, 1989.
- [15] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," in *Proc. Conf. Computer Vision*, 1987, pp. 34–54.
- [16] R. Bajcsy, "Active perception," *Proc. IEEE*, vol. 76, no. 8, pp. 996–1005, Aug. 1988.
- [17] E. Krotkov and R. Bajcsy, "Active vision for reliable ranging: cooperating focus, stereo, and vergence," *Int. J. Comput. Vis.*, vol. 11, no. 2, pp. 187–203, 1993.
- [18] G. Sandini, F. Gandolfo, E. Grosso, and M. Tistarelli, "Vision during action," in *Active Perception*, Y. Aloimonos, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1993.
- [19] R. D. Rimey and C. Brown, "Control of selective perception using bayes nets and decision theory," *Int. J. Comput. Vis.*, vol. 12, no. 2/3, pp. 173–207, 1994.
- [20] N. Oliver and E. Horvitz, "Selective perception policies for guiding sensing and computation in multimodal systems: a comparative analysis," in *Proc. 5th ICML*, Washington, DC, Aug. 2003.
- [21] A. Pinz, M. Prantl, H. Ganster, and H. K. Borotschnig, "Active fusion—a new method applied to remote sensing image interpretation," *Pattern Recognit. Lett.*, vol. 17, pp. 1349–1359, 1996.
- [22] L. Paletta and A. Pinz, "Active object recognition by view integration and reinforcement learning," *Robot. Auton. Syst.*, no. 31, pp. 71–86, 2000.
- [23] J. Denzler and C. M. Brown, "Information theoretic sensor data selection for active object recognition and state estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 145–157, Feb/ 2002.
- [24] C. Soyer, H. I. Bozma, and Y. İstefanopoulos, "Attentional sequence-based recognition: Markovian and evidential reasoning," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 33, no. 6, pp. 937–950, Dec. 2003.
- [25] A. Logothetis and A. Isaksson, "On sensor scheduling via information theoretic criteria," in *Proc. American Control Conf.*, San Diego, CA, 1999, pp. 2402–2406.
- [26] J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *EURASIP J. Appl. Signal Process.*, vol. 2003, no. 4, 2003.
- [27] E. Ertin, J. Fisher, and L. Potter, "Maximum mutual information principle for dynamic sensor query," in *Proc. IPSN'03*, Palo Alto, CA, 2003.
- [28] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based sensor selection heuristic for target localization," in *Proc. IPSN'04*, Berkeley, CA, USA, 2004.
- [29] V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden Markov model sensors," *IEEE Trans. Signal Process.*, vol. 50, no. 6, pp. 1382–1397, Jun. 2002.
- [30] S. Jiang, R. Kumar, and H. E. Garcia, "Optimal sensor selection for discrete-event systems with partial observation," *IEEE Trans. Automat. Contr.*, vol. 48, no. 3, pp. 369–381, Mar. 2002.
- [31] I. Bloch, "Information combination operators for data fusion: A comparative review with classification," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 26, no. 1, pp. 52–67, Jan. 1996.
- [32] S. L. Lauritzen and N. Wermuth, "Graphical models for association between variables, some of which are qualitative and some quantitative," *Ann. Statist.*, vol. 17, pp. 31–57, 1989.
- [33] M. Frydenberg, "The chain graph Markov property," *Scand. J. Statist.*, vol. 17, pp. 333–353, 1990.
- [34] W. L. Buntine, "Chain graphs for learning," in *Proc. 11th Conf. UAI*, San Francisco, CA, 1995.
- [35] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.
- [36] P. S. Maybeck, *Stochastic Models, Estimation, and Control*. New York: Academic, 1979.
- [37] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
- [38] Z. Elouedi, K. Mellouli, and P. Smets, "Assessing sensor reliability for multisensor data fusion within the transferable belief model," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 782–787, Feb. 2004.
- [39] R. L. Keeney and H. Raiffa, *Decision With Multiple Objectives: Preferences and Value Tradeoffs*. Cambridge, U.K.: Cambridge Univ. Press, 1993.
- [40] Y. Zhang and Q. Ji, "Sensor selection for active information fusion," in *Proc. 20th Nat. Conf. Artificial Intelligence*, Pittsburgh, PA, 2005.

Structure Learning of Bayesian Networks using Constraints

Cassio P. de Campos

CASSIO@IDSIA.CH

Dalle Molle Institute for Artificial Intelligence (IDSIA), Galleria 2, Manno 6928, Switzerland

Zhi Zeng

ZENGZ@RPI.EDU

Qiang Ji

JIQ@RPI.EDU

Rensselaer Polytechnic Institute (RPI), 110 8th St., Troy NY 12180, USA

Abstract

This paper addresses exact learning of Bayesian network structure from data and expert's knowledge based on score functions that are decomposable. First, it describes useful properties that strongly reduce the time and memory costs of many known methods such as hill-climbing, dynamic programming and sampling variable orderings. Secondly, a branch and bound algorithm is presented that integrates parameter and structural constraints with data in a way to guarantee global optimality with respect to the score function. It is an any-time procedure because, if stopped, it provides the best current solution and an estimation about how far it is from the global solution. We show empirically the advantages of the properties and the constraints, and the applicability of the algorithm to large data sets (up to one hundred variables) that cannot be handled by other current methods (limited to around 30 variables).

1. Introduction

A Bayesian network (BN) is a probabilistic graphical model that relies on a structured dependency among random variables to represent a joint probability distribution in a compact and efficient manner. It is composed by a directed acyclic graph (DAG) where nodes are associated to random variables and conditional probability distributions are defined for variables given their parents in the graph. Learning the graph (or structure) of a BN from data is one of the

most challenging problems in such models. Best exact known methods take exponential time on the number of variables and are applicable to small settings (around 30 variables). Approximate procedures can handle larger networks, but usually they get stuck in local maxima. Nevertheless, the quality of the structure plays a crucial role in the accuracy of the model. If the dependency among variables is not properly learned, the estimated distribution may be far from the *correct* one. In general terms, the problem is to find the best structure (DAG) according to some score function that depends on the data (Heckerman et al., 1995). There are other approaches to learn a structure that are not based on scoring (for example taking some statistical similarity among variables), but we do not discuss them in this paper. The research on this topic is active, e.g. (Chickering, 2002; Teyssier & Koller, 2005; Tsamardinos et al., 2006). Best exact ideas (where it is guaranteed to find the global best scoring structure) are based on dynamic programming (Koivisto et al., 2004; Singh & Moore, 2005; Koivisto, 2006; Silander & Myllymaki, 2006), and they spend time and memory proportional to $n \cdot 2^n$, where n is the number of variables. Such complexity forbids the use of those methods to a couple of tens of variables, mostly because of memory consumption.

In the first part of this paper, we present some properties of the problem that bring a considerable improvement on many known methods. We perform the analysis over some well known criteria: *Akaike Information Criterion* (AIC), and the *Minimum Description Length* (MDL), which is equivalent to the *Bayesian Information Criterion* (BIC). However, results extrapolate to the Bayesian Dirichlet (BD) scoring (Cooper & Herskovits, 1992) and some derivations under a few assumptions. We show that the search space of possible structures can be reduced drastically without losing the global optimality guarantee and that the memory requirements are very small in many practical cases

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

(we show empirically that only a few thousand scores are stored for a problem with 50 variables and one thousand instances).

As data sets with many variables cannot be efficiently handled (unless $P=NP$, as the problem is known to be NP-hard (Chickering et al., 2003)), a desired property of a method is to produce an *any-time* solution, that is, the procedure, if stopped at any moment, provides an approximate solution, while if run until it finishes, a global optimum solution is found. However, the most efficient exact methods are not *any-time*. We propose a new any-time exact algorithm using a branch-and-bound (B&B) approach with caches. Scores are computed during the initialization and a poll is built. Then we perform the search over the possible graphs iterating over arcs. Although iterating over orderings is probably faster, iterating over arcs allows us to work with constraints in a straightforward way. Because of the B&B properties, the algorithm can be stopped at any-time with a best current solution found so far and an upper bound to the global optimum, which gives a kind of certificate to the answer and allows the user to stop the computation when she believes that the current solution is good enough. (Suzuki, 1996) has proposed a B&B method, but it is not a global exact algorithm, instead the search is conducted after a node ordering is fixed. Our method does not rely on a pre-defined ordering and finds a global optimum structure considering all possible orderings.

2. Bayesian networks

A BN represents a single joint probability density over a collection of random variables. It can be defined as a triple $(\mathcal{G}, \mathcal{X}, \mathcal{P})$, where $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ is a DAG with $V_{\mathcal{G}}$ a collection of n nodes associated to random variables \mathcal{X} (a node per variable), and $E_{\mathcal{G}}$ a collection of arcs; \mathcal{P} is a collection of conditional probability densities $p(X_i|PA_i)$ where PA_i denotes the parents of X_i in the graph (PA_i may be empty), respecting the relations of $E_{\mathcal{G}}$. We assume throughout that variables are categorical. In a BN every variable is conditionally independent of its non-descendants given its parents (Markov condition). This structure induces a joint probability distribution by the expression $p(X_1, \dots, X_n) = \prod_i p(X_i|PA_i)$. Before proceeding, we define some notations. Let $r_i \geq 2$ be the number of discrete categories of X_i , q_i the number of elements in Ω_{PA_i} (the number of configurations of the parent set, that is, $q_i = \prod_{X_t \in PA_i} r_t$) and θ be the entire vector of parameters such as $\theta_{ijk} = p(x_i^k|pa_i^j)$, where $i \in \{1, \dots, n\}$, $j \in \{1, \dots, q_i\}$, $k \in \{1, \dots, r_i\}$ (hence $x_i^k \in \Omega_{X_i}$ and $pa_i^j \in \Omega_{PA_i}$).

Given a complete data set $D = \{D_1, \dots, D_N\}$ of with N instances, with $D_t = \{x_{1,t}^{k_1}, \dots, x_{n,t}^{k_n}\}$ a instance of all variables, the goal of structure learning is to find a \mathcal{G} that maximizes a score function such as MDL or AIC.

$$\max_{\mathcal{G}} s_D(\mathcal{G}) = \max_{\theta} (L_D(\theta) - t \cdot W),$$

where θ represents all parameters of the model (and thus depends on the graph \mathcal{G}), $t = \sum_{i=1}^n (q_i \cdot (r_i - 1))$ is the number of free parameters, W is criterion-specific ($W = \frac{\log N}{2}$ in MDL and $W = 1$ in AIC), and L_D is the log-likelihood function:

$$L_D(\theta) = \log \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}}, \quad (1)$$

where n_{ijk} indicates how many elements of D contain both x_i^k and pa_i^j . This function can be written as $L_D(\theta) = \sum_{i=1}^n L_{D,i}(\theta_i)$, where $L_{D,i}(\theta_i) = \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} n_{ijk} \log \theta_{ijk}$. From now on, the subscript D is omitted for simplicity.

An important property of such criteria is that they are decomposable, that is, they can be applied to each node X_i separately: $\max_{\mathcal{G}} s(\mathcal{G}) = \max_{\mathcal{G}} \sum_{i=1}^n s_i(PA_i)$, where $s_i(PA_i) = L_i(PA_i) - t_i(PA_i) \cdot W$, with $L_i(PA_i) = \max_{\theta_i} L_i(\theta_i)$ (θ_i is the parameter vector related to X_i , so it depends on the choice of PA_i), and $t_i(PA_i) = q_i \cdot (r_i - 1)$. Because of this property and to avoid computing such functions several times, we create a cache that contains $s_i(PA_i)$ for each X_i and each parent set PA_i . Note that this cache may have an exponential size on n , as there are 2^{n-1} subsets of $\{X_1, \dots, X_n\} \setminus \{X_i\}$ to be considered as parent sets. This gives a total space and time of $O(n \cdot 2^n)$ to build the cache. Instead, the following results show that this number is much smaller in many practical cases.

Lemma 1 *Let X_i be a node of \mathcal{G}' , a DAG for a BN where $PA_i = J'$. Suppose $J \subset J'$ is such that $s_i(J) > s_i(J')$. Then J' is not the parent set of X_i in the optimal DAG.*

Proof. Take a graph \mathcal{G} that differs from \mathcal{G}' only on $PA_i = J$, which is also a DAG (as the removal of some arcs does not create cycles) and $s(\mathcal{G}) = \sum_{j \neq i} s_j(PA_j) + s_i(J) > \sum_{j \neq i} s_j(PA_j) + s_i(J') = s(\mathcal{G}')$. Hence any DAG \mathcal{G}' such that $PA_i = J'$ has a subgraph \mathcal{G} with a better score than \mathcal{G}' , and thus J' is not the optimal parent configuration for X_i . \square

Lemma 1 is quite simple but very useful to discard elements from the cache of X_i . However, it does not tell anything about supersets of J' , that is, we still need to compute all the possible parent configurations

and later verify which of them can be removed. Next theorems handle this issue.

Theorem 1 *Using MDL or AIC as score function and assuming $N \geq 4$, take \mathcal{G} and \mathcal{G}' DAGs such that \mathcal{G} is a subgraph of \mathcal{G}' . If \mathcal{G} is such that $\prod_{j \in PA_i} r_j \geq N$, for some X_i , and X_i has a proper superset of parents in \mathcal{G}' w.r.t. \mathcal{G} , then \mathcal{G}' is not an optimal structure.*

Proof.¹ Take a DAG \mathcal{G} such that $J = PA_i$ for a node X_i , and take \mathcal{G}' equal to \mathcal{G} except that it contains an extra node in $J^{new} = PA_i$, that is, in \mathcal{G}' we have $J^{new} = J \cup \{X_e\}$. Note that the difference in the scores of the two graphs are restricted to $s_i(\cdot)$. In the graph \mathcal{G}' , $L_i(J^{new})$ will certainly not decrease and $t_i(J^{new})$ will increase, both with respect to the values for \mathcal{G} . The difference in the scores will be $s_i(J^{new}) - s_i(J)$, which equals to

$$\begin{aligned} & L_i(J^{new}) - t_i(J^{new}) - (L_i(J) - t_i(J)) \leq \\ & - \sum_{j=1}^{q_i} \sum_{i=1}^{r_i} n_{ijk} \log \theta_{ijk} - t_i(J^{new}) + t_i(J) \leq \\ & \sum_{j=1}^{q_i} n_{ij} \left(- \sum_{i=1}^{r_i} \frac{n_{ijk}}{n_{ij}} \log \frac{n_{ijk}}{n_{ij}} \right) - t_i(J^{new}) + t_i(J) \leq \\ & \sum_{j=1}^{q_i} n_{ij} H(\theta_{ij}) - t_i(J^{new}) + t_i(J) \leq \\ & \sum_{j=1}^{q_i} n_{ij} \log r_i - q_i \cdot (r_e - 1) \cdot (r_i - 1) \cdot W \end{aligned}$$

The first step uses the fact that $L_i(J^{new})$ is negative, the second step uses that fact that $\hat{\theta}_{ijk} = \frac{n_{ijk}}{n_{ij}}$, with $n_{ij} = \sum_{i=1}^{r_i} n_{ijk}$, is the value that maximizes $L_i(\cdot)$, and the last step uses the fact that the entropy of a discrete distribution is less than the log of its number of categories. Finally, \mathcal{G} is a better graph than \mathcal{G}' if the last equation is negative, which happens if $q_i \cdot (r_e - 1) \cdot (r_i - 1) \cdot W \geq N \log r_i$. Because $r_i \geq 2 \Rightarrow r_i - 1 \geq \log r_i$, and $N \geq 4 \Rightarrow \frac{\log N}{2} \geq 1$ (the W of the MDL case), we have that $q_i = \prod_{j \in J} r_j \geq N$ ensures that $s_i(J^{new}) < s_i(J)$, which implies that the graph \mathcal{G}' cannot be optimal. \square

Corollary 1 *In the optimal structure \mathcal{G} , each node has at most $O(\log N)$ parents.*

Proof. It follows directly from Theorem 1 and the fact that $r_i \geq 2$, for all X_i . \square

Theorem 1 and Corollary 1 ensures that the cache stores at most $O\left(\frac{n-1}{\log N}\right)$ elements for each variable

¹Another similar proof appears in (Bouckaert, 1994), but it leads directly to the conclusion of Corollary 1. The intermediate result is algorithmically important.

(all combinations up to $\log N$ parents). Although it does not help us to improve the theoretical size bound, Lemma 2 gives us even less elements.

Lemma 2 *Let X_i be a node with $J \subset J'$ two possible parent sets such that $t_i(J') + s_i(J) > 0$. Then J' and all supersets $J'' \supset J'$ are not optimal parent configurations for X_i .*

Proof. Because $L_i(\cdot)$ is a negative function, $t_i(J') + s_i(J) > 0 \Rightarrow -t_i(J') - s_i(J) < 0 \Rightarrow (L_i(J') - t_i(J')) - s_i(J) < 0 \Rightarrow s_i(J') < s_i(J)$. Using Lemma 1, we have that J' is not the optimal parent set for X_i . The result also follows for any $J'' \supset J$, as we know that $t_i(J'') > t_i(J')$. \square

Thus, the idea is to check the validity of Lemma 2 every time the score of a parent set J' of X_i is about to be computed, discarding J' and all supersets whenever possible. This result allows us to stop computing scores for J' and all its supersets. Lemma 1 is stronger, but regards a comparison between exactly two parent configuration. Nevertheless, Lemma 1 can be applied to the final cache to remove all certainly useless parent configurations. As we see in Section 5, the practical size of the cache after these properties is small even for large networks. Lemma 1 is also valid for other decomposable functions, including BD and derivations (e.g. BDe, BDeu), so the benefits shall apply to those scores too, and the memory requirements will be reduced. The other theorems need assumptions about the initial N and the choice of priors. Further discussion is left for future work because of lack of space.

3. Constraints

An additional way to reduce the space of possible DAGs is to consider some constraints provided by experts. We work with two main types of constraints: constraints on parameters that define rules about the probability values inside the local distributions of the network, and structural constraints that specify where arcs may or may not be included.

3.1. Parameter Constraints

We work with a general definition of parameter constraint, where any convex constraint is allowed. If θ_{i,PA_i} is the parameter vector of the node X_i with parent set PA_i , then a *convex constraint* is defined as $h(\theta_{i,PA_i}) \leq 0$, where $h : \Omega_{\theta_{i,PA_i}} \rightarrow \mathcal{R}$ is a convex function over θ_{i,PA_i} . This definition includes many well known constraints, for example from Qualitative Probabilistic Networks (QPN) (Wellman, 1990): *qualitative influences* define some knowledge about the state of

a variable given the state of another, which roughly means that observing a greater state for a parent X_a of a variable X_b makes more likely to have greater states in X_b (for any parent configuration except for X_a). For example, $\theta_{b_j22} \geq \theta_{b_j12}$, where $j_k \doteq \{x_a^k, pa_b^{j*}\}$ and j_* is an index ranging over all parent configurations except for X_a . In this case, observing x_a^2 makes more likely to have x_b^2 . A *negative influence* is obtained by replacing the inequality operator \geq by \leq , and a *zero influence* is obtained by changing inequality to an equality. Other constraints such as *synergies* (Wellman, 1990) are also linear and local to a single node.

Although we allow the parameter constraints that are general, we have the following restriction about them: if a constraint is specified for a node X_i and a set of parents J , then the actual parent set PA_i has to be a superset of J . Furthermore, we have a peculiar interpretation for each constraint C as follows: if $J \subset PA_i$ (proper subset), then the parameter constraint must hold for all configurations of the parents of X_i that do not belong to J . For example, suppose X_1 has X_2 and X_3 as parents (all of them binary), and the following constraint h was defined on X_1 : $p(x_1^2|x_2^2x_3^2) + 2 \cdot p(x_1^2|x_2^2x_3^1) \leq 1$. If a new node X_4 is included as parent of X_1 , the constraint h becomes the two following constraints:

$$\begin{aligned} p(x_1^2|x_2^2x_3^2x_4^1) + 2 \cdot p(x_1^2|x_2^2x_3^1x_4^1) &\leq 1, \\ p(x_1^2|x_2^2x_3^2x_4^2) + 2 \cdot p(x_1^2|x_2^2x_3^1x_4^2) &\leq 1, \end{aligned}$$

that is, h holds for each state of X_4 . For example if another parent X_5 is included, then four constraints would be enforced with all possible combinations. This interpretation for constraints is in line with the definition of qualitative constraints of QPNs, and most importantly, it allows us to treat the constraints in a principled way for each set of parents. It means that the constraint must hold for all configurations of parents not involved in the constraint, which can be also interpreted as *other parents are not relevant and the constraint is valid for each one of their configurations*.

3.2. Structural constraints

Besides probabilistic constraints, we work with structural constraints on the possible graphs. These constraints help to reduce the search space and are available in many situations. We work with the following rules:

- $indegree(X_j, k, op)$, where $op \in \{lt, eq\}$ and k an integer, means that the node X_j must have *less than* (when $op = lt$) or *equal to* (when $op = eq$) k parents.

- $arc(X_i, X_j)$ indicates that the node X_i must be a parent of X_j .
- Operators *or* (\vee) and *not* (\neg) are used to form the rules. The *and* operator is not explicitly used as we assume that each constraint is in disjunctive normal form.

For example, the constraints $\forall_{i \neq c, j \neq c} \neg arc(X_i, X_j)$ and $indegree(X_c, 0, eq)$ impose that only arcs from node X_c to the others are possible, and that X_c is a root node, that is, a Naive Bayes structure will be learned. The procedure will also act as a feature selection procedure by letting some variables unlinked. Note that the symbol \forall just employed is not part of the language but is used for easy of expose (in fact it is necessary to write down every constraint defined by such construction). As another example, the constraints $\forall_{j \neq c} indegree(X_j, 3, lt)$, $indegree(X_c, 0, eq)$, and $\forall_{j \neq c} indegree(X_j, 0, eq) \vee arc(X_c, X_j)$ ensure that all nodes have X_c as parent, or no parent at all. Besides X_c , each node may have at most one other parent, and X_c is a root node. This learns the structure of a Tree-augmented Naive (TAN) classifier, also performing a kind of feature selection (some variables may end up unlinked). In fact, it learns a forest of trees, as we have not imposed that all variables must be linked.

3.3. Dealing with constraints

All constraints in previous examples can be imposed during the construction of the cache, because they involve just a single node each. In essence, parent sets of a node X_i that do violate some constraint are not stored in the cache, and this can be checked during the cache construction. On the other hand, constraints such as $arc(X_1, X_2) \vee arc(X_2, X_3)$ cannot be imposed in that stage, as they impose a non-local condition (the arcs go to distinct variables, namely X_2 and X_3), because the cache construction is essentially a local procedure with respect to each variable. Such constraints that involve distinct nodes can be verified during the B&B phase, so they are addressed later.

Regarding parameter constraints, we compute the scores using a constrained optimization problem, i.e. maximize the score function subject to simplex equality constraints and all parameter constraints defined by the user.

$$\begin{aligned} \max_{\theta_i} L_i(\theta_i) - t_i(PA_i) \\ \text{subject to } \quad &\forall_{j=1 \dots q_i} g_{ij}(\theta_{ij}) = 0, \quad (2) \\ &\forall_{z=1 \dots m_{hi}} h_{iz}(\theta_i) \leq 0, \end{aligned}$$

where $g_{ij}(\theta_{ij}) = -1 + \sum_{k=1}^{r_i} \theta_{ijk}$ imposes that distributions defined for each variable given a parent configura-

tion sum one over all variable states, and the m_{hi} convex constraints h_{iz} define the space of feasible parameters for the node X_i . This is possible because: (1) we have assumed that a constraint over $p(x_i^k | x_{i_1}^{k_1}, \dots, x_{i_t}^{k_t})$ forces $X_{i_1}, \dots, X_{i_t} \subseteq PA_i$, that is, when a parameter constraint is imposed, the parent set of the node must contain at least the variables involved in the constraint; (2) the optimization is computed for every possible parent set, that is, PA_i is known in the moment to write down the optimization problem, which is solved for each X_i and each set PA_i . We use the optimization package of (Birgin et al., 2000).

Theorem 2 *Using MDL or AIC as score function and assuming $N \geq 4$, take \mathcal{G} and \mathcal{G}' as DAGs such that \mathcal{G} is a subgraph of \mathcal{G}' . Suppose that both \mathcal{G} and \mathcal{G}' respect the same set of parameter and structural constraints. If \mathcal{G} is such that $\prod_{j \in PA_i} r_j \geq N$, for some X_i , and X_i has a proper superset of parents in \mathcal{G}' w.r.t. \mathcal{G} , then \mathcal{G}' is not an optimal structure.*

Proof. Just note that all derivations in Theorem 1 are also valid in the case of constraints. The only difference that deserves a comment is $\hat{\theta}_{ijk} = \frac{n_{ijk}}{n_{ij}}$, which may be an unfeasible point for the optimization (2), because the latter contains parameter constraints that might reduce the parameter space (besides the normal constraints of the maximum log-likelihood problem). As $\hat{\theta}_{ijk}$ is just used as an upper value for the log-likelihood function, and the constrained version can just obtain smaller objective values than the unconstrained version, $\frac{n_{ijk}}{n_{ij}}$ is an upper bound also for the constrained case. Thus, the derivation of Theorem 1 is valid even with constraints. \square

Corollary 1 and Lemmas 1 and 2 are also valid in this setting. The proof of Corollary 1 is straightforward, as it only depends on Theorem 1, while for Lemmas 1 and 2 we need just to ensure that all the parent configurations that are discussed there respect the constraints.

4. Constrained B&B algorithm

In this section we describe the B&B algorithm used to find the best structure of the BN and comment on its complexity, correctness, and some extensions and particular cases. The notation (and initialization of the algorithm) is as follows: $C : (X_i, PA_i) \rightarrow \mathcal{R}$ is the cache with the scores for all the variables and their possible parent configurations (using Theorem 1 and Lemmas 1 and 2 to have a reduced size); \mathcal{G} is the graph created taking the best parent configuration for each node without checking for acyclicity (so it is not necessarily a DAG), and s is the score of \mathcal{G} ; \mathcal{H} is an initially empty matrix containing, for each possible arc

between nodes, a mark stating that the arc must be present, or is prohibited, or is free (may be present or not); Q is a priority queue of triples $(\mathcal{G}, \mathcal{H}, s)$, ordered by s (initially it contains a single triple with \mathcal{G} , \mathcal{H} and s just mentioned; and finally $(\mathcal{G}_{best}, s_{best})$ is the best DAG and score found so far (s_{best} is initialized with $-\infty$). The main loop is as follows:

While Q is not empty, do

1. Remove the peek $(\mathcal{G}_{cur}, \mathcal{H}_{cur}, s_{cur})$ of Q . If $s \leq s_{best}$ (worse than an already known solution), then start the loop again. If \mathcal{G}_{cur} is a DAG and satisfies all structural constraints, update $(\mathcal{G}_{best}, s_{best})$ with $(\mathcal{G}_{cur}, s_{cur})$ and start the loop again.
2. Take $v = (X_{a_1} \rightarrow X_{a_2} \rightarrow \dots \rightarrow X_{a_{q+1}})$, with $a_1 = a_{q+1}$, is a directed cycle of \mathcal{G}_{cur} .
3. For $y = 1, \dots, q$, do
 - Mark on \mathcal{H}_{cur} that the arc $X_{a_y} \rightarrow X_{a_{y+1}}$ is prohibited.
 - Recompute (\mathcal{G}, s) from $(\mathcal{G}_{cur}, s_{cur})$ such that the parents of $X_{a_{y+1}}$ in \mathcal{G} comply with this restriction and with \mathcal{H}_{cur} . Furthermore, the subgraph of \mathcal{G} formed by arcs that are demanded by \mathcal{H}_{cur} (those that have a mark *must exist*) must comply with the structural constraints (it might be impossible to get such graph. In such case, go to the last bullet). Use the values in the cache $C(X_{a_{y+1}}, PA_{a_{y+1}})$ to avoid recomputing scores.
 - Include the triple $(\mathcal{G}, \mathcal{H}_{cur}, s)$ into Q .
 - Mark on \mathcal{H}_{cur} that the arc $X_{a_y} \rightarrow X_{a_{y+1}}$ must be present and that the sibling arc $X_{a_{y+1}} \rightarrow X_{a_y}$ is prohibited, and continue.

The algorithm uses a B&B search where each case to be solved is a relaxation of a DAG, that is, they may contain cycles. At each step, a graph is picked up from a priority queue, and it is verified if it is a DAG. In such case, it is a feasible structure for the network and we compare its score against the best score so far (which is updated if needed). Otherwise, there must be a directed cycle in the graph, which is then broken into subcases by forcing some arcs to be absent/present. Each subcase is put in the queue to be processed. The procedure stops when the queue is empty. Note that every time we break a cycle, the subcases that are created are independent, that is, the sets of graphs that respect \mathcal{H} for each subcase are disjoint. We obtain this fact by properly breaking the cycles: when $v = (X_{a_1} \rightarrow X_{a_2} \rightarrow \dots \rightarrow X_{a_{q+1}})$ is detected, we create q

subcases such that the first does not contain $X_{a_1} \rightarrow X_{a_2}$ (but may contain the other arcs of that cycle), the second case certainly contains $X_{a_1} \rightarrow X_{a_2}$, but $X_{a_2} \rightarrow X_{a_3}$ is prohibited (so they are disjoint because of the difference in the presence of the first arc), and so on such that the y -th case certainly contains $X_{a_{y'}} \rightarrow X_{a_{y'+1}}$ for all $y' < y$ and prohibits $X_{a_y} \rightarrow X_{a_{y+1}}$. This idea ensures that we never process the same graph twice. So the algorithm runs at most $\prod_i |C(X_i)|$ steps, where $|C(X_i)|$ is the size of the cache for X_i .

B&B can be stopped at any time and the current best solution as well as an upper bound for the global best score are available. This stopping criterion might be based on the number of steps, time and/or memory consumption. Moreover, the algorithm can be easily parallelized. We can split the content of the priority queue into many different tasks. No shared memory needs to exist among tasks if each one has its own version of the cache. The only data structure that needs consideration is the queue, which from time to time must be balanced between tasks. With a message-passing idea that avoids using process locks, the gain of parallelization is linear in the number of tasks. As far as we know, best known exact methods are not easily parallelized, they do not deal with constraints, and they do not provide lower and upper estimates of the best structure if stopped early. If run until it ends, the proposed method gives a global optimum solution for the structure learning problem.

Some particular cases of the algorithm are worth mentioning. If we fix an ordering for the variables such that all the arcs must link a node towards another non-precedent in the ordering (this is a common idea in many approximate methods), the proposed algorithm does not perform any branch, as the ordering implies acyclicity, and so the initial solution is already the best. The performance would be proportional to the time to create the cache. On the other hand, bounding the maximum number of parents of a node is relevant only for hardest inputs, as it would imply a bound on the cache size, which is already empirically small.

5. Experiments

We perform experiments to show the benefits of the reduced cache and search space and the gains of constraints.² First, we use data sets available at the UCI repository (Asuncion & Newman, 2007). Lines with missing data are removed and continuous variables are discretized over the mean into binary variables. The

²The software is available online through the web address <http://www.ecse.rpi.edu/~cvrl/structlearning.html>

data sets are: *adult* (15 variables and 30162 instances), *car* (7 variables and 1728 instances) *letter* (17 variables and 20000 instances), *lung* (57 variables and 27 instances), *mushroom* (23 variables and 1868 instances), *nursery* (9 variables and 12960 instances), *Wisconsin Diagnostic Breast Cancer* or *wdbc* (31 variables and 569 instances), *zoo* (17 variables and 101 instances). No constraints are employed in this phase as we intend to show the benefits of the properties earlier discussed.

Table 1 presents the cache construction results, applying Theorem 1 and Lemmas 1 and 2. Its columns show the data set name, the number of steps the procedure spends to build the cache (a step equals to a call to the score function for a single variable and a parent configuration), the time in seconds, the size of the generated cache (number of scores stored, the memory consumption is actually $O(n)$ times that number), and finally the size of the cache if all scores were computed. Note that the reduction is huge. Although in the next we are going to discuss three distinct algorithms, the benefits of the application of these results imply in performance gain for other algorithms in the literature to learn BN structures. It is also possible to analyze the search space reduction implied by these results by looking columns 2 and 3 of Table 2.

Table 1. Cache sizes (number of stored scores) and time (in seconds) to build them for many networks and data sizes. Steps represent the number of local (single node given a parent set) score evaluations.

name	steps	time(s)	size	$n2^n$
adult	30058	182.09	672	$2^{17.9}$
car	335	0.09	24	$2^{8.8}$
letter	534230	2321.46	41562	$2^{20.1}$
lung	43592	1.33	3753	$2^{61.8}$
mushroom	140694	72.13	8217	$2^{26.5}$
nursery	1905	3.94	49	$2^{11.2}$
wdbc	1692158	351.04	7482	2^{35}
zoo	9118	0.31	1875	$2^{20.1}$

In Table 2, we show results of three distinct algorithms: the B&B described in Section 4, the dynamic programming (DP) idea of (Silander & Myllymaki, 2006), and an algorithm that picks variable orderings randomly and then find the best structure such that all arcs link a node towards another that is not precedent in the ordering. This last algorithm (named OS) is similar to K2 algorithm with random orderings, but it is always better because a global optimum is found for each ordering.³ The scores obtained by each algorithm (in percentage against the value obtained by B&B) and

³We have run a hill-climbing approach (which is also benefited by ideas presented in this paper), but its accuracy was worse than OS. We omit it because of lack of space.

Table 2. Comparison of MDL scores among B&B, dynamic programming (DP), and ordering sampling (one thousand times). *Fail* means that it could not solve the problem within 10 million steps. DP and OS scores are in percentage w.r.t. the score of B&B (positive percentage means worse than B&B and negative percentage means better).

network	search	reduced	B&B			DP		OS	
	space	space	score	gap	time(s)	score	time(s)	score	time(s)
adult	2^{210}	2^{71}	-286902.8	5.5%	150.3	0.0%	0.77	0.1%	0.17
car	2^{42}	2^{10}	-13100.5	0.0%	0.01	0.0%	0.01	0.0%	0.01
letter	2^{272}	2^{188}	-173716.2	8.1%	574.1	-0.6%	22.8	1.0%	0.75
lung	2^{3192}	2^{330}	-1146.9	2.5%	907.1	<i>Fail</i>	<i>Fail</i>	1.0%	0.13
mushroom	2^{506}	2^{180}	-12834.9	15.3%	239.8	<i>Fail</i>	<i>Fail</i>	1.0%	0.12
nursery	2^{72}	2^{17}	-126283.2	0.0%	0.04	0.0%	0.04	0.0%	0.04
wdbc	2^{930}	2^{216}	-3053.1	13.6%	333.5	<i>Fail</i>	<i>Fail</i>	0.8%	0.13
zoo	2^{272}	2^{111}	-773.4	0.0%	5.2	0.0%	3.5	1.0%	0.03

the corresponding spent time are presented (excluding the cache construction). A limit of ten million steps is given to each method (steps here are considered as the number of queries to the cache). It is also presented the reduced space where B&B performs its search, as well as the maximum gap of the solution. This gap is obtained by the relaxed version of the problem. So we can guarantee that the global optimal solution is within this gap (even though the solution found by the B&B may already be the best, as shown in the first line of the table). With the reduced cache presented here, finding the best structure for a given ordering is very fast, so it is possible to run OS over millions of orderings in a short period of time.

Table 3. B&B procedure learning TANs. Time (in seconds) to find the global optimum, cache size (number of stored scores) and (reduced) space for B&B search.

network	time(s)	cache size	space
adult	0.26	114	2^{39}
car	0.01	14	$2^{6.2}$
letter	0.32	233	2^{61}
lung	0.26	136	2^{51}
mushroom	0.71	398	2^{88}
nursery	0.06	26	2^{12}
wdbc	361.64	361	2^{99}

Some additional comments are worth. DP can solve the *mushroom* set in less than 10 minutes if we drop the limit of steps. The expectation for *wdbc* is around four days. Hence, we cannot expect to obtain an answer in larger cases, such as *lung*. It is clear that, in worst case, the number of steps of DP is smaller than that of B&B. Nevertheless, B&B eventually bounds some regions without processing them, provides an upper bound at each iteration, and does not suffer from memory exhaustion as DP. This makes the method applicable even to very large settings. Still, DP seems a good choice for small n . When n is large (more than 35), DP will not finish in reasonable time, and hence

will not provide any solution, while B&B still gives an approximation and a bound to the global optimum. About OS, if we sample one million times instead of one thousand as done before, its results improve and the global optimum is found also for *adult* and *mushroom* sets. Still, OS provides no guarantee or estimation about how far is the global optimum (here we know it has achieved the optimum because of the exact methods). It is worth noting that both DP and OS are benefited by the smaller cache.

Table 3 shows the results when we employ constraints to force the final network to be a Tree-augmented Naive Bayes (*zoo* was run, but it is not included because the unconstrained learned network was already TAN). Here the class is isolated in the data set and constraints are included as described in Section 3.2. Note that the cache size, the search space and consequently the time to solve the problems have all decreased. Finally, Table 4 has results for random data sets with predefined number of nodes and instances. A randomly created BN with at most $3n$ arcs is used to sample the data. Because of that, we are able to generate random parameter and structural constraints that are certainly valid for this *true* BN (approximately $n/2$ constraints for each case). The table contains the total time to run the problem and the size of the cache, together with the percentage of gain when using constraints. Note that the code was run in parallel with a number of tasks equals to n , otherwise an increase by a factor of n must be applied to the results in the table. We can see that the gain is recurrent in all cases (the constrained version has also less gap in all cases, although such number is not shown).

6. Conclusions

This paper describes a novel algorithm for learning BN structure from data and expert’s knowledge. It integrates structural and parameter constraints with

Table 4. Results on random data sets generated from random networks. Time to solve (10 million steps) and size of the cache are presented for the *normal* unconstrained case and the percentage of gain when using constraints.

nodes(n)/ instances	unconstrained			constrained gain	
	gap	time(s)	cache	time	cache
30/100	0%	0.06	125	67%	11.6%
30/500	0%	2.7	143	47.5%	26.5%
50/100	0%	0.26	310	31.4%	16.1%
50/500	0%	20.66	231	57.2%	29.8%
70/100	0%	4.58	1205	36.9%	18.8%
70/500	1.1%	356.9	666	38.4%	21.9%
100/100	0.5%	9.05	2201	47.5%	23.5%
100/500	1.4%	1370.4	726	50.2%	33.0%

data through a B&B procedure that guarantees global optimality with respect a decomposable score function. It is an any-time procedure in the sense that, if stopped early, it provides the best current solution found so far and a maximum error of such solution. The software is available as described in the experiments.

We also describe properties of the structure learning problem based on scoring DAGs that enable the B&B procedure presented here as well as other methods to work over a reduced search space and memory. Such properties allow the construction of a cache with all possible local scores of nodes and their parents without large memory consumption.

Because of the properties and the characteristics of the B&B method, even without constraints the B&B is more efficient than state-of-the-art exact methods for large domains. We show through experiments with randomly generated data and public data sets that problems with up to 70 nodes can be exactly processed in reasonable time, and problems with 100 nodes are handled within a small worst case error. These results surpass by far current methods, and may also help to improve other approximate methods and may have interesting practical applications, which we will pursue in future work.

Acknowledgments

This work is supported in part by the grant W911NF-06-1-0331 from the U.S. Army Research Office. The first author thanks also the project *Ticino in Rete*.

References

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.

Birgin, E. G., Martínez, J. M., & Raydan, M. (2000).

Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. on Optimiz.*, 10, 1196–1211.

- Bouckaert, R. (1994). Properties of bayesian belief network learning algorithms. *Conf. on Uncertainty in Artificial Intelligence* (pp. 102–109). M. Kaufmann.
- Chickering, D., Meek, C., & Heckerman, D. (2003). Large-sample learning of bayesian networks is np-hard. *Conf. on Uncertainty in Artificial Intelligence* (pp. 124–13). M. Kaufmann.
- Chickering, D. M. (2002). Optimal structure identification with greedy search. *J. of Machine Learning Research*, 3, 507–554.
- Cooper, G., & Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Mach. Learning*, 9, 309–347.
- Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Mach. Learning*, 20, 197–243.
- Koivisto, M. (2006). Advances in exact bayesian structure discovery in bayesian networks. *Conf. on Uncertainty in Artificial Intelligence* (pp. 241–248) AUAI Press.
- Koivisto, M., Sood, K., & Chickering, M. (2004). Exact bayesian structure discovery in bayesian networks. *J. of Machine Learning Research*, 5, 2004.
- Silander, T., & Myllymaki, P. (2006). A simple approach for finding the globally optimal bayesian network structure. *Conf. on Uncertainty in Artificial Intelligence*. (pp. 445–452) AUAI Press.
- Singh, A. P., & Moore, A. W. (2005). *Finding optimal bayesian networks by dynamic programming* (Technical Report). Carnegie Mellon Univ. CALD-05-106.
- Suzuki, J. (1996). Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the B&B technique. *Int. Conf. on Machine Learning* (pp. 462–470).
- Teyssier, M., & Koller, D. (2005). Ordering-based search: A simple and effective algorithm for learning bayesian networks. *Conf. on Uncertainty in Artificial Intelligence*. (pp. 584–590) AUAI Press.
- Tsamardinos, I., Brown, L. E., & Aliferis, C. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Mach. Learning*, 65, 31–78.
- Wellman, M. P. (1990). Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44, 257–303.



Learning Bayesian network parameters under incomplete data with domain knowledge

Wenhui Liao^{a,*}, Qiang Ji^b

^aThomson Reuters, Eagan, MN 55123, USA

^bECSE, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

ARTICLE INFO

Article history:

Received 24 April 2008

Received in revised form 4 February 2009

Accepted 7 April 2009

Keywords:

Bayesian network parameter learning

Missing data

EM algorithm

Facial action unit (AU) recognition

ABSTRACT

Bayesian networks (BNs) have gained increasing attention in recent years. One key issue in Bayesian networks is parameter learning. When training data is incomplete or sparse or when multiple hidden nodes exist, learning parameters in Bayesian networks becomes extremely difficult. Under these circumstances, the learning algorithms are required to operate in a high-dimensional search space and they could easily get trapped among copious local maxima. This paper presents a learning algorithm to incorporate domain knowledge into the learning to regularize the otherwise ill-posed problem, to limit the search space, and to avoid local optima. Unlike the conventional approaches that typically exploit the quantitative domain knowledge such as prior probability distribution, our method systematically incorporates qualitative constraints on some of the parameters into the learning process. Specifically, the problem is formulated as a constrained optimization problem, where an objective function is defined as a combination of the likelihood function and penalty functions constructed from the qualitative domain knowledge. Then, a gradient-descent procedure is systematically integrated with the E-step and M-step of the EM algorithm, to estimate the parameters iteratively until it converges. The experiments with both synthetic data and real data for facial action recognition show our algorithm improves the accuracy of the learned BN parameters significantly over the conventional EM algorithm.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

In recent years, Bayesian networks (BNs) have been increasingly used in a wide range of applications including computer vision [1], bioinformatics [2], information retrieval [3], data fusion [4], decision support systems and others. A BN is a directed acyclic graph (DAG) that represents a joint probability distribution among a set of variables, where the nodes denote random variables and the links denote the conditional dependencies among variables. The advantages of BNs can be summarized as their semantic clarity and understandability by humans, the ease of acquisition and incorporation of prior knowledge, the possibility of causal interpretation of learned models, and the automatic handling of noisy and missing data [5].

In spite of these claims, people often face the problem of learning BNs from training data in order to apply BNs to real-world applications. Typically, there are two categories in learning BNs, one is to learn BN parameters when a BN structure is known, and another is

to learn both BN structures and parameters. In this paper, we focus on BN parameter learning by assuming the BN structure is already known. If the training data is complete, learning BN parameters is not difficult, however, in real world, training data can be incomplete for various reasons. For example, in a BN modeling video surveillance, the training data may be incomplete because of security issue; in a BN modeling customer behaviors, the training data may be incomplete because of privacy issue. Sometimes, the training data may be complete but sparse, because some events rarely happen, or the data for these events are difficult to obtain.

In general, training data can be missed in three ways: *missing at random (MAR)*, *missing completely at random (MCAR)*, and *not missing at random (NMAR)*. MAR means the probability of missing data on any variable is not related to its particular value, but could be related to other variables. MCAR means the missing value of a variable depends neither on the variable itself nor on the values of other variables in the BN. For example, some hidden (latent) nodes never have data. NMAR means data missing is not at random but depends on the values of the variables.

The majority of the current learning algorithms assume the MAR property holds for all the incomplete training samples since learning is easier for MAR than NMAR and MCAR. The classical approaches

* Corresponding author.

E-mail addresses: wenhui.liao@thomsonreuters.com (W. Liao), qji@ecse.rpi.edu (Q. Ji).

include the Expectation Maximization (EM) algorithm [6] and Gibbs sampling [7]. Other methods are proposed to overcome the disadvantages of EM and Gibbs sampling. For example, methods are proposed to learn the parameters when data are not missing at random, such as the AI&M procedure [8], the RBE algorithm [9], and the maximum entropy method [10,11]; some methods are proposed to escape local maxima under the assumption of MAR, such as the information-bottleneck EM (IB-EM) algorithm [12], data perturbation method [13], etc.; other methods are proposed to speed up the learning procedure, such as generalized conjugate gradient algorithm [14], online updating rules [15], and others.

When data are missing completely at random, in other words, when several hidden nodes exist, those methods could fail, where the learned parameters may be quite different from the true parameters. In fact, since there are no data for hidden nodes, learning parameters becomes an ill-posed problem. Thus, prior data on domain knowledge are needed to regularize the learning problem. In most domains, at least some information, either from literature or from domain experts, is available about the model to be constructed. However, many forms of prior knowledge that an expert might have are difficult to be directly used by existing machine learning algorithms. Therefore, it is important to formalize the knowledge systematically and incorporate it into the learning. Such domain knowledge can help regularize the otherwise ill-posed learning problem, reduce the search space significantly, and help escape local maxima.

This motivates us to propose a Bayesian network learning algorithm for the case when multiple hidden nodes exist by systematically combining domain knowledge during learning. Instead of using quantitative domain knowledge, which is often hard to obtain, we propose to exploit qualitative domain knowledge. Qualitative domain knowledge impose approximated constraints on some parameters or on the relationships among some parameters. These kind of qualitative knowledge are often readily available. Specifically, two qualitative constraints are considered, the range of parameters, and the relative relationships between different parameters. Instead of using the likelihood function as the objective to maximize during learning, we define the objective function as a combination of the likelihood function and the penalty functions constructed from the domain knowledge. Then, a gradient-descent procedure is systematically integrated with the Expectation-step (E-step) and Maximization-step (M-step) of the EM algorithm, to estimate the parameters iteratively until it converges. The experiments show the proposed algorithm significantly improves the accuracy of the learned BN parameters over the conventional EM method.

2. Related work

During the past several years, many methods have been proposed to learn BN parameters when data are missing. Two standard learning algorithms are Gibbs sampling [7] and EM [6]. Gibbs sampling by Geman and Geman [7] is the basic tool of simulation and can be applied to virtually any graphical model whether the arcs are directed or not, and whether the variables are continuous or discrete [16]. It completes the samples by inferring the missing data from the available information and then learns from the completed database (imputation strategy). Unfortunately, Gibbs sampling method suffers from convergence problems arising from correlations between successive samples [10]. In addition, it is not effective when data are missing in complete random (e.g. the case of the hidden nodes).

The EM algorithm can be regarded as a deterministic version of Gibbs sampling used to search for the Maximum Likelihood (ML) or Maximum a Posteriori (MAP) estimate for model parameters [16,6]. However, when there are multiple hidden variables or a large amount of missing data, EM gets easily trapped in a local maximum. "With

data missing massively and systematically, the likelihood function has a number of local maxima and straight maximum likelihood gives results with unsuitably extreme probabilities" [17]. In addition, EM algorithms are sensitive to the initial starting points. If the initial starting points are far away from the optimal solution, the learned parameters are not reliable.

Different methods are proposed to help avoid local maxima. Elidan and Friedman [12] propose an information-bottleneck EM (IB-EM) algorithm to learn the parameters of BNs with hidden nodes. It treats the learning problem as a tradeoff between two information-theoretic objectives, where the first one is to make the hidden nodes uninformative about the identity of specific instances, and the second one is to make the hidden variables informative about the observed attributes. However, although IB-EM has a better performance than the standard EM for some simple BNs, it is actually worse than EM for the complex hierarchical models as shown in [12]. To escape local maxima in learning, Elida et al. [13] propose a solution by perturbing training data. Two basic techniques are used to perturb the weights of the training data: (1) random reweighing, which randomly samples weight profiles on the training data, and (2) adversarial reweighing, which updates the weight profiles to explicitly punish the current hypothesis, with the intent of moving the search quickly to a nearby basin of attraction. Although it usually achieves better solutions than EM, it is still a heuristic method and not necessarily able to escape local maxima. And also, it is much slower than the standard EM algorithm.

The previous methods emphasize improving the machine learning techniques, instead of using domain knowledge to help learning. Since there are no data available for hidden nodes, it is important to incorporate any available information about these nodes into learning. The methods for constraining the parameters for a BN include Dirichlet priors, parameter sharing, and qualitative constraints. According to [18], there are several problems using Dirichlet priors. First, it is impossible to represent even the simple equality constraints on the parameters. Second, it is often beyond expert's capability to specify a full Dirichlet prior over the parameters of a Bayesian network. Parameter sharing, on the other hand, allows parameters of different models to share the same values, i.e., it allows to impose equality constraints. Parameter sharing methods, however, do not capture more complicated constraints among parameters such as inequality constraints among the parameters. In addition, both Dirichlet priors and parameter sharing methods are restricted to sharing parameters at the level of sharing a whole CPT or CPTs, instead of at the level of granularity of individual parameters. To overcome these limitations, others [19–22,18] propose to explicitly exploit qualitative relationships among parameters and systematically incorporate them into the parameter estimation process.

Druzdel et al. [19] give formal definitions of several types of qualitative relationships that can hold between nodes in a BN to help specify CPTs of BNs, including probability intervals, qualitative influences, and qualitative synergies. They express these available information in a canonical form consisting of (in)equalities expressing constraints on the hyperspace of possible joint probability distributions, and then use this canonical form to derive upper and lower bounds on any probability of interest. However, the upper and lower bounds cannot give sufficient insight into how likely a value from the interval is to be the actual probability.

Wittig and Jameson [20] present a method for integrating formal statements of qualitative constraints into two learning algorithms, APN [23,24] and EM. Two types of qualitative influences [19] are considered as constraints for parameters during learning in this method: (1) a positive influence holds between two variables (X_1, X_2) if for any given value of X_2 , an increase in the value of X_1 will not decrease the probability that the value of X_2 is equal to or greater than that given value; and (2) a negative influence can be defined analogously.

This paper shows that the accuracy of the learned BNs is superior to that of the corresponding BNs learned without the constraints.

Even when data are complete, domain knowledge can help learning significantly, in particular when insufficient data are available. Altendorf et al. [21] show how to interpret knowledge of qualitative influences, and in particular of monotonicities, as constraints on probability distribution, and to incorporate this knowledge into BN learning algorithms. It assumes that the values of the variables can be totally ordered. It focuses on learning from complete but sparse data. It shows the additional constraints provided by qualitative monotonicities can improve the performance of BN classifiers, particularly on extremely small training sets. The assumption that the values of the variables can be totally ordered is, however, too restrictive.

Feelders and Gaag [22,25] present a method for learning BN parameters with prior knowledge about the signs of influences between variables. The various signs are translated into order constraints on the network parameters and isotonic regression is then used to compute order-constrained estimates from the available data. They also focus only on learning from complete but sparse data. In addition, they assume all variables are binary.

The constraints analyzed in these papers [20–22] are somewhat restrictive, in the sense that each constraint must involve all parameters in a conditional probability table (CPT). Niculescu [18] considers a different type of domain knowledge for constraining parameter estimate when data are complete but limited. Two types of constraints are used: one is that the sum of several parameters within one conditional probability distribution is bounded by the sum of other parameters in the same distribution; another is an upper bound on the sum of several parameters in one conditional probability distribution. Both the constraints have to be twice differentiable, with continuous second derivatives. They formalize the learning as a constraint optimization problem and derive closed form maximum likelihood parameter estimators.

Our learning method also exploits domain knowledge to help parameter learning, but especially for BNs with hidden nodes or a large amount of missing data. Different from other learning methods, the domain knowledge we used has the following features: (1) it does not need to involve all the parameters in a conditional probability table; (2) it can deal with relationships between different parameters; (3) it is associated with confidence levels to reflect the confidence of the domain experts; (4) it is easy to assess and define; and (5) new format of domain knowledge can be easily incorporated into the algorithm. Our algorithm systematically incorporates the domain knowledge and is capable of learning parameters successfully even when a large percent of nodes are hidden in a Bayesian network.

3. Learning Bayesian network parameters

3.1. The basic theory

Let G be a BN with nodes X_1, \dots, X_n . If there is a directed arc from X_i to X_j , X_i is called a parent of X_j , $pa(X_j)$. Given its parent nodes, a node is conditionally independent from all the other nodes. Thus the joint distribution of all the nodes can be written as

$$p(X_1, \dots, X_n) = \prod_{i=1}^n p(X_i | pa(X_i)) \quad (1)$$

Each node is associated with several parameters to describe the conditional probability distribution of the random variable given its parents. We use θ to denote the entire vector of parameter value θ_{ijk} , $\theta_{ijk} = p(x_i^k | pa_i^j)$, where i ($i = 1, \dots, n$) ranges over all the variables in the BN, j ($j = 1, \dots, q_i$) ranges over all the possible parent configurations of X_i , and k ($k = 1, \dots, r_i$) ranges over all the possible states of X_i . Therefore, x_i^k represents the k th state of variable X_i , and pa_i^j is the j th configuration of the parent nodes of X_i .

Given the data set $D = \{D_1, \dots, D_N\}$, where $D_l = \{x_1[l], \dots, x_n[l]\}$ that consists of instances of the BN nodes, the goal of parameter learning is to find the most probable value $\hat{\theta}$ for θ that can best explain the data set D , which is usually quantified by the log-likelihood function, $\log(p(D|\theta))$, denoted as $L_D(\theta)$. If D is complete, based on the conditional independence assumption in BNs as well as the assumptions that the samples are independent, we can get the equation as follows:

$$\begin{aligned} L_D(\theta) &= \log \left\{ \prod_{m=1}^N p(x_1[m], \dots, x_n[m] : \theta) \right\} \\ &= \log \left\{ \prod_{i=1}^n \prod_{m=1}^N p(x_i[m] | pa_i[x_i(m)] : \theta) \right\} \end{aligned} \quad (2)$$

where $pa_i[x_i(m)]$ indicates the i th parent of $x_i(m)$. With the MLE (Maximum Likelihood Estimation) method, we can get the parameter θ^* as follows:

$$\theta^* = \arg \max_{\theta} L_D(\theta) \quad (3)$$

However, when the data D is incomplete, Eq. (2) cannot be directly applied anymore. A common method is the EM algorithm [6]. Let $Y = \{Y_1, Y_2, \dots, Y_N\}$, which is observed data; $Z = \{Z_1, Z_2, \dots, Z_N\}$, which is missing data; and $D_l = Y_l \cup Z_l$. The EM algorithm starts with some initial guess at the maximum likelihood parameter, $\theta^{(0)}$, and then proceeds to iteratively generate successive estimates, $\theta^{(1)}, \theta^{(2)}, \dots$, by repeatedly applying the Expectation-step and Maximization-step, for $t = 1, 2, \dots$

E-step: computes the conditional expectation of the log-likelihood function given the observed data Y and the current parameter $\theta^{(t)}$

$$Q(\theta | \theta^{(t)}) = E_{\theta^{(t)}} [\log p(D|\theta) | \theta^{(t)}, Y] \quad (4)$$

M-step: finds a new parameter $\theta^{(t+1)}$ which maximizes the expected log-likelihood under the assumption that the distribution found in the E-step is correct

$$\theta^{(t+1)} = \arg \max_{\theta} Q(\theta | \theta^{(t)}) \quad (5)$$

Each iteration is guaranteed to increase the likelihood, and finally the algorithm converges to a local maximum of the likelihood function. However, when there are multiple hidden nodes or when a large amount of data are missing, EM method easily gets stuck in a local maximum. Next, we show how to incorporate domain knowledge into the learning process in order to reduce search space and to avoid local maxima.

3.2. Qualitative constraints with confidence

In many real-world applications, domain experts usually have valuable information about model parameters. We consider two types of constraints: type-I is about the range of a parameter; and type-II is about the relative relationships ($>$, $<$, $=$) between different parameters. One of our goals is to make the constraints as simple as possible, so that experts can easily formalize their knowledge into these constraints.

The range of a parameter allows domain experts to specify an upper bound and a lower bound for the parameter, instead of defining specific values. Fig. 1 shows a very simple BN and we assume all the nodes have binary states. The table in the right is the conditional probability table of the node B , indicating $P(B|A)$, which can be described by four parameters $\theta_{B00}, \theta_{B01}, \theta_{B10}, \theta_{B11}$, where $\theta_{B00} + \theta_{B01} = 1$ and $\theta_{B10} + \theta_{B11} = 1$. The domain experts may not know the specific

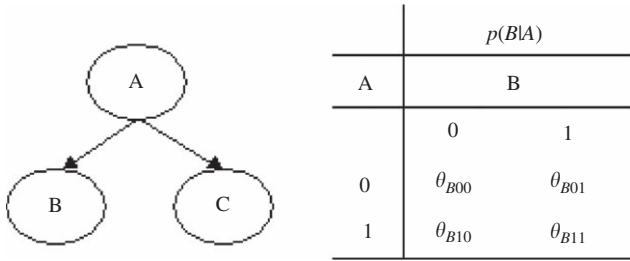


Fig. 1. A simple Bayesian network example and its conditional probability table.

values of θ_{B00} and θ_{B10} , but they can set the ranges for θ_{B00} and θ_{B10} as $0.3 < \theta_{B00} < 0.5$, $0.6 < \theta_{B10} < 0.9$, respectively.

In addition to assessing the ranges of parameters, the domain experts may also know the relative relationships between some parameters. For each type-II constraint, if the two associated parameters are in the same CPT of a node, we call it an inner-relationship constraint; if the two parameters come from the CPTs of different nodes, we call it an outer-relationship constraint. For example, if the observation of $A=0$ increases the posterior of $B=0$ the most, we could say that $p(B=0|A=0) > p(B=0|A=1)$, in other words, $\theta_{B00} > \theta_{B01}$. Obviously, the constraint of $\theta_{B00} > \theta_{B01}$ defines an inner-relationship. On the other hand, if the observation of $A=0$ increases the posterior of $B=0$ more than the observation of $A=0$ increasing the posterior of $C=0$, we can get the constraint of $\theta_{B00} > \theta_{C00}$, which defines an outer-relationship since B and C are from two different CPTs.

In real-world applications, there are always such constraints that can be found by domain experts. For example, assume we use a BN to model human state and its symptoms such as blinking rate, head tilt rate, eye gaze distribution, etc. Some symptom may be a stronger indicator of a particular human state than another symptom. This kind of relationship can be captured by the type-II constraints in the BN. They can often be identified either through subjective experience or through a sensitivity analysis. These constraints look simple, but are very important for the hidden nodes, where no data are available. By adding these constraints into learning, the domain knowledge can be well utilized to obtain parameters that meet the requirements of real-world applications.

Now we formally define the two types of constraints. Let A be the set that includes the parameters whose ranges are known based on the domain knowledge. For each $\theta_{ijk} \in A$, we define the range as $l_{ijk} \leq \theta_{ijk} \leq u_{ijk}$. Obviously, $l_{ijk} \geq 0$, and $u_{ijk} \leq 1$. Let B be the set that includes the parameters whose relative relationships are known based on the domain knowledge. For each $\theta_{ijk}, \theta_{i'j'k'} \in B$, we have $\theta_{ijk} > \theta_{i'j'k'}$, and/or, $\theta_{ijk} = \theta_{i'j'k'}$, where $i \neq i'$, or $j \neq j'$, or $k \neq k'$.

However, the domain knowledge may not be reliable all the time. To account for it, we associate confidence levels $\lambda_{ijk}, \lambda_{i'j'k'}$ to each constraint in the sets A and B , respectively. The value of each λ is between 0 and 1. If a domain expert is very confident with a constraint, the corresponding value of λ is 1; otherwise, λ is less than 1 but larger than 0.

3.3. Parameter learning with uncertain qualitative constraints

Now our goal is to find the optimal parameter $\hat{\theta}$ that maximizes the log-likelihood $L_D(\theta)$ given the three constraints as below:

$$\begin{aligned} & \text{Maximize } L_D(\theta) \\ & \text{Subject to } \sum_k \theta_{ijk} = 1 \\ & \quad 1 \leq i \leq n, \quad 1 \leq j \leq q_i, \quad 1 \leq k \leq q_i \\ & \quad l_{ijk} \leq \theta_{ijk} \leq u_{ijk}, \quad \theta_{ijk} \in A \\ & \quad \theta_{ijk} \geq \theta_{i'j'k'}, \theta_{ijk}, \theta_{i'j'k'} \in B \end{aligned} \tag{6}$$

The above equation shows a constrained optimization problem. For each inequality constraint, we define the following penalty functions:

$$g'(\theta_{ijk}) = [\theta_{ijk} - l_{ijk}]^-, \quad \forall \theta_{ijk} \in A \tag{7}$$

$$g''(\theta_{ijk}) = [u_{ijk} - \theta_{ijk}]^-, \quad \forall \theta_{ijk} \in A \tag{8}$$

$$g'''(\theta_{ijk}, \theta_{i'j'k'}) = [\theta_{ijk} - \theta_{i'j'k'}]^- , \quad \forall \theta_{ijk}, \theta_{i'j'k'} \in B \tag{9}$$

where $[x]^- = \max(0, -x)$.

Therefore, we can rephrase Eq. (6) as follows:

$$\begin{aligned} & \text{Maximize } J(\theta) = L_D(\theta) - \frac{w_1}{2} \sum_{\theta_{ijk} \in A} \lambda_{ijk} [(g'(\theta_{ijk}))^2 + (g''(\theta_{ijk}))^2] \\ & \quad - \frac{w_2}{2} \sum_{\theta_{ijk}, \theta_{i'j'k'} \in B} \lambda_{ijk}^{i'j'k'} (g'''(\theta_{ijk}, \theta_{i'j'k'}))^2 \\ & \text{Subject to } \sum_k \theta_{ijk} = 1 \end{aligned} \tag{10}$$

where w_i is the penalty weight, which is decided empirically. Obviously, the penalty varies with the confidence level for each constraint.

In order to solve the problem, first, we eliminate the constraint $\sum_k \theta_{ijk} = 1$ by reparameterizing θ_{ijk} , so that the new parameters automatically respect the constraint on θ_{ijk} no matter what their values are. We define a new parameter β_{ijk} so that

$$\theta_{ijk} \equiv \frac{\exp(\beta_{ijk})}{\sum_{k'=1}^{q_i} \exp(\beta_{ijk'})} \tag{11}$$

In this way, a local maximum w.r.t. to β_{ijk} is also a local maximum w.r.t. θ_{ijk} , and vice versa. Most importantly, the constraint is automatically satisfied.

In the next step, we need to compute the derivative of $J(\theta)$ w.r.t. β . Based on [24], $\nabla_{\theta_{ijk}} L_D(\theta)$ can be expressed as follows:

$$\begin{aligned} \nabla_{\theta_{ijk}} L_D(\theta) &= \frac{\partial \ln \prod_{l=1}^N p(D_l|\theta)}{\partial \theta_{ijk}} \\ &= \sum_{l=1}^N \frac{\partial \ln p(D_l|\theta)}{\partial \theta_{ijk}} \\ &= \sum_{l=1}^N \frac{\partial \ln p(D_l|\theta) / \partial \theta_{ijk}}{p(D_l|\theta)} \end{aligned} \tag{12}$$

where

$$\begin{aligned} \frac{\partial \ln p(D_l|\theta) / \partial \theta_{ijk}}{p(D_l|\theta)} &= \frac{\frac{\partial}{\partial \theta_{ijk}} \sum_{j',k'} p(D_l|x_i^{j'}, p\alpha_i^{j'}, \theta) p(x_i^{j'} | p\alpha_i^{j'}, \theta)}{p(D_l|\theta)} \\ &= \frac{\frac{\partial}{\partial \theta_{ijk}} \sum_{j',k'} p(D_l|x_i^{j'}, p\alpha_i^{j'}, \theta) p(x_i^{j'} | p\alpha_i^{j'}, \theta) p(p\alpha_i^{j'} | \theta)}{p(D_l|\theta)} \\ &= \frac{p(D_l|x_i^k, p\alpha_i^k, \theta) p(p\alpha_i^k | \theta)}{p(D_l|\theta)} \\ &= \frac{p(x_i^k, p\alpha_i^k | D_l, \theta) p(D_l|\theta) p(p\alpha_i^k | \theta)}{p(x_i^k, p\alpha_i^k | \theta) p(D_l|\theta)} \\ &= \frac{p(x_i^k, p\alpha_i^k | D_l, \theta)}{p(x_i^k, p\alpha_i^k | \theta)} \\ &= \frac{p(x_i^k, p\alpha_i^k | D_l, \theta)}{\theta_{ijk}} \end{aligned} \tag{13}$$

By combining Eqs. (12) and (13), we obtain the equation as below:

$$\nabla_{\theta_{ijk}} L_D(\theta) = \frac{\sum_{l=1}^N p(x_i^k, p\alpha_i^l | D_l, \theta)}{\theta_{ijk}} \quad (14)$$

Therefore, based on the chain rule,

$$\begin{aligned} \nabla_{\beta_{ijk}} L_D(\theta) &= \frac{\partial L_D(\theta)}{\partial \theta_{ijk}} \frac{\partial \theta_{ijk}}{\partial \beta_{ijk}} \\ &= \nabla_{\theta_{ijk}} L_D(\theta) (\theta_{ijk} - \theta_{ijk}^2) \\ &= \sum_{l=1}^N p(x_i^k, p\alpha_i^l | D_l, \theta) (1 - \theta_{ijk}) \end{aligned} \quad (15)$$

Similarly, for $g'(\theta_{ijk})$, $g''(\theta_{ijk})$, and $g'''(\theta_{ijk})$, the derivatives are as follows:

$$\nabla_{\beta_{ijk}} g'(\theta_{ijk}) = \begin{cases} \theta_{ijk}^2 - \theta_{ijk} & \text{if } \theta_{ijk} \leq l_{ijk} \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

$$\nabla_{\beta_{ijk}} g''(\theta_{ijk}) = \begin{cases} \theta_{ijk} - \theta_{ijk}^2 & \text{if } \theta_{ijk} \geq u_{ijk} \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

$$\nabla_{\beta_{ijk}} g'''(\theta_{ijk}, \theta_{i'j'k'}) = \begin{cases} \theta_{ijk}^2 - \theta_{ijk} & \text{if } \theta_{ijk} \leq \theta_{i'j'k'}, i \neq i' \text{ or } j \neq j' \\ \theta_{ijk}^2 - \theta_{ijk} - \theta_{ijk} \theta_{i'j'k'} & \text{if } \theta_{ijk} \leq \theta_{i'j'k'}, i = i', j = j' \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

Table 1
A constrained EM (CEM) learning algorithm.

Repeat until it converges Step 1: E-step to compute the conditional expectation of the log-likelihood function based on Eq. (4); Step 2: M-step to find the parameter θ^t that maximizes the expected log-likelihood based on Eq. (5); Step 3: Perform the following optimization procedure based on the gradient-descent method: $\theta^t = \theta^t$; map θ^t to β^t based on Eq. (11) Repeat until $\Delta\beta^t \approx 0$ $\Delta\theta^t = 0$ for each variable i , parent configuration j , value k for each $D_l \in D$ $\Delta\theta_{ijk}^{t+1} = \Delta\theta_{ijk}^t + p(x_i^k, p\alpha_i^l D_l, \theta_t)$ $\Delta\beta_{ijk}^{t+1} = \Delta\theta_{ijk}^{t+1} (1 - \theta_{ijk}^t) + K$ (K represents the last three terms in Eq. (19)) $\beta^{t+1} = \beta^t + \Delta\beta^{t+1}$ map β^{t+1} to θ^{t+1} based on Eq. (11) $\theta^t = \theta^{t+1}$ Go to Step 1 Return θ^t
--

Therefore, the derivative of $J(\theta_{ijk})$ w.r.t. β_{ijk} is as we can see here:

$$\begin{aligned} \nabla_{\beta_{ijk}} J(\theta_{ijk}) &= \sum_{l=1}^N p(x_i^k, p\alpha_i^l | D_l, \theta) (1 - \theta_{ijk}) \\ &\quad - w_1 \lambda_{ijk} [g'(\theta_{ijk}) \nabla_{\beta_{ijk}} g'(\theta_{ijk}) + g''(\theta_{ijk}) \nabla_{\beta_{ijk}} g''(\theta_{ijk})] \\ &\quad - w_2 \sum_{B^+(\theta_{ijk})} [\lambda_{ijk}^{i'j'k'} g'''(\theta_{ijk}, \theta_{i'j'k'}) \nabla_{\beta_{ijk}} g'''(\theta_{ijk}, \theta_{i'j'k'})] \\ &\quad + w_2 \sum_{B^-(\theta_{ijk})} [\lambda_{ijk}^{i'j'k'} g'''(\theta_{i'j'k'}, \theta_{ijk}) \nabla_{\beta_{ijk}} g'''(\theta_{i'j'k'}, \theta_{ijk})] \end{aligned} \quad (19)$$

where $B^+(\theta_{ijk})$ is the set of the constraints whose first term is θ_{ijk} , while $B^-(\theta_{ijk})$ is the set of the constraints whose second term is θ_{ijk} . Both $B^+(\theta_{ijk})$ and $B^-(\theta_{ijk})$ belong to the set B .

Now, we are ready to present the constrained EM (CEM) learning algorithm as summarized in Table 1. The algorithm consists of three steps. The first two steps are the same as the E-step and M-step in the EM algorithm. In the third step, a gradient-based update is used to force the solutions to move towards the direction of reducing constraint violations.

4. Experiments

In this section, we compare our algorithm to the EM algorithm with synthetic data. We first describe how the testing data is generated, and then demonstrate the results in three scenarios: (1) varying the type-I constraints; (2) varying the type-II constraints; and (3) varying the number of training samples. Furthermore, in the next section, we will apply our algorithm to a real-world application.

4.1. Experiment design

In order to compare the CEM algorithm to the EM algorithm, we design the experiments as follows.

Generation of original BNs. Two BNs as shown in Fig. 2 are created in the experiments. Then 22 instances are created with the same structures as BN1 and BN2, respectively (11 for each), but with different parameters. Two BNs (called original BNs) are then used as the ground-truth for BN1 and BN2, respectively, and the others are to be learned.

Generation of constraints. For each BN, based on the true CPTs, type-I (the range of a parameter) and type-II (the relationship between different parameters) constraints are generated for the node sets A and B , where A and B vary in the experiments. Specifically, to generate type-I constraints for the nodes in A , for each true parameter θ_{ijk} , the lower bound is set as $(1 - r)\theta_{ijk}$, and the upper bound is set as $\min(1, (1 + r)\theta_{ijk})$, where r is a ratio ($0 < r < 1$) and varies in the experiments. Type-II constraints can be divided into

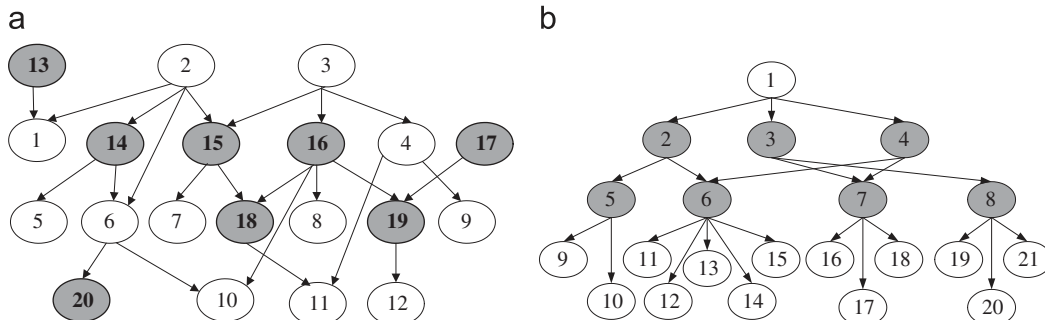


Fig. 2. Two BN examples: (a) BN1; (b) BN2. The shaded nodes are hidden nodes, the others are observable nodes.

two kinds: inner-relationship, which compares two parameters within the same CPT; outer-relationship, which compares two parameters in different CPTs. For example, in BN1, two parameters in the CPT of the node 18 can be associated with an inner-relationship

Table 2
Some examples of type-II constraints for BN1.

Inner-relationship constraints	Outer-relationship constraints
Node 14: $\theta_{1411} > \theta_{1421}$	Node 13 and Node 17: $\theta_{1311} > \theta_{1711}$
Node 15: $\theta_{1511} > \theta_{1522}$	Node 14 and Node 15: $\theta_{1411} > \theta_{1531}$
Node 20: $\theta_{2011} > \theta_{2021}$	Node 18 and Node 19: $\theta_{1821} > \theta_{1941}$

constraint; while a parameter in the CPT of the node 18 and a parameter in the CPT of the node 19 can be associated with an outer-relationship constraint. Table 2 shows some examples of type-II constraints for BN1. For each parameter θ_{abcd} in the table, the first two numbers (ab) of the subscript represent the index of the node, and the third number (c) represents the index of the parent configurations, and the last number (d) represents the state index of the node.

Generation of training data. For each BN, 500 samples are generated based on the true parameters. The values of the hidden nodes are then removed from the generated samples. With the remaining samples, we then learn the 20 BNs with randomly assigned initial parameters, which are required to be different from the true parameters.

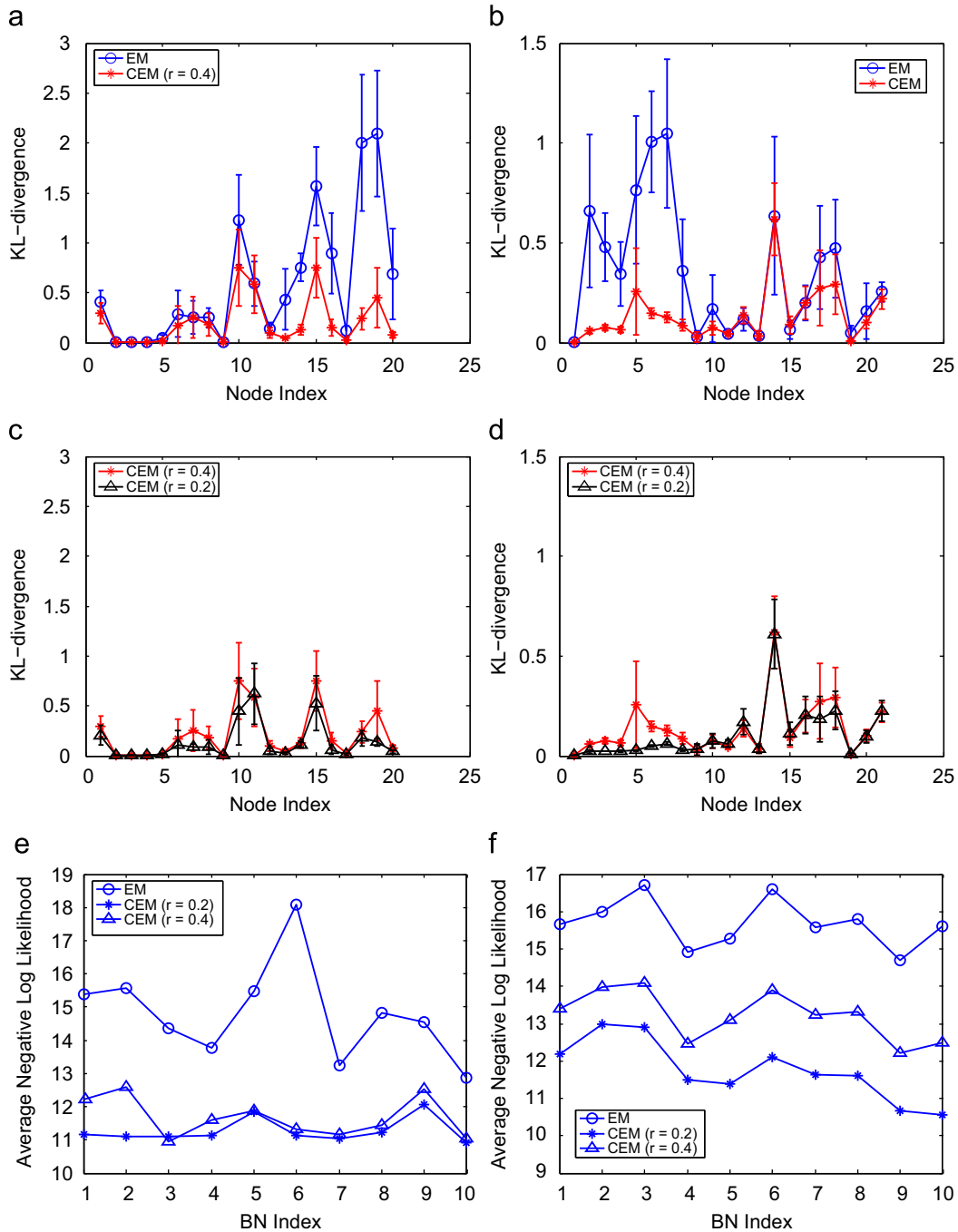


Fig. 3. Learning results vs. type-I constraints. The charts in the left are the results for BN1; and the charts in the right are the results for BN2. (a), (b) EM vs. CEM when $r = 0.4$; (c), (d) CEM when $r = 0.4$ and 0.2 ; and (e), (f) negative log-likelihood for different BNs.

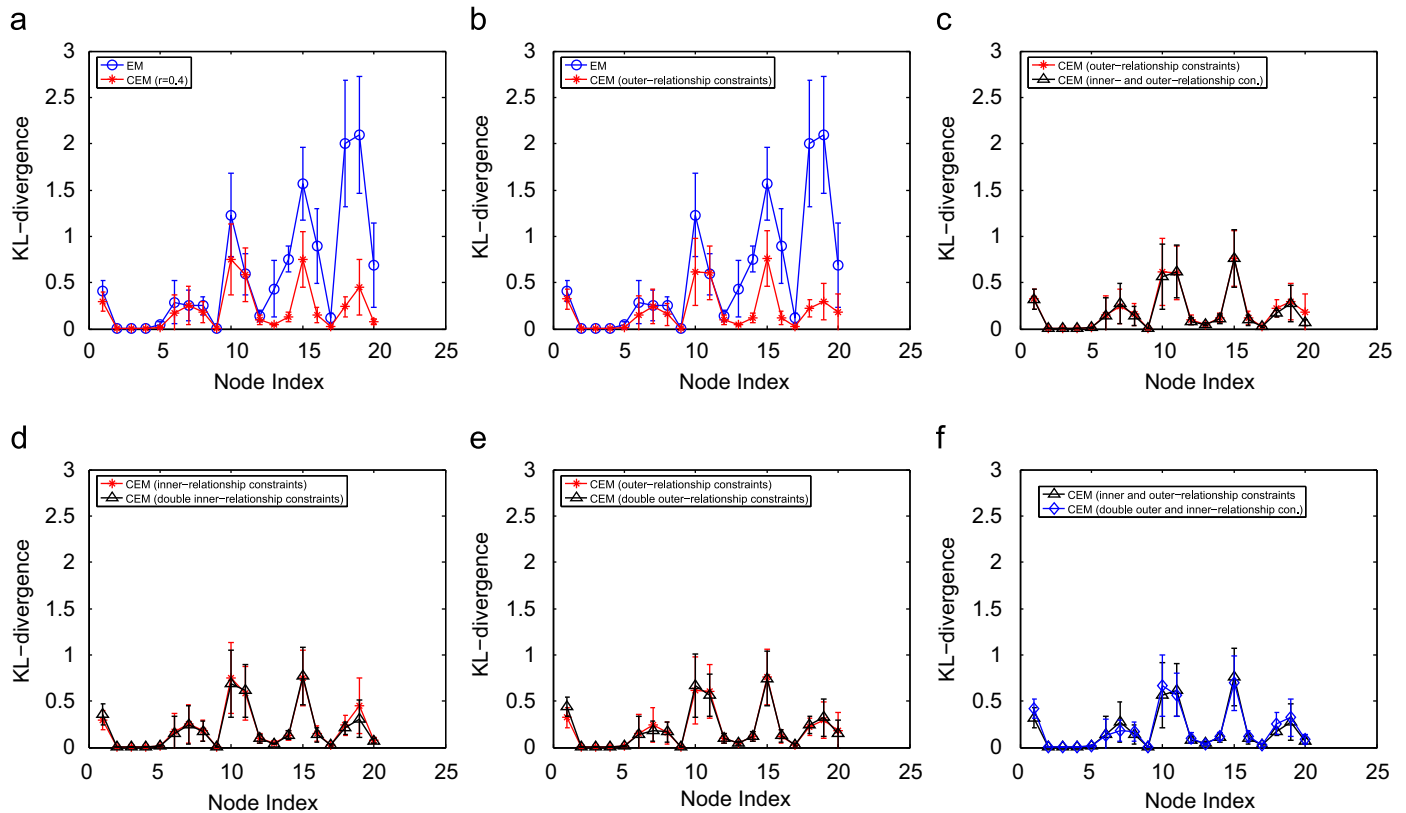


Fig. 4. Learning results vs. type-II constraints for BN1: (a) EM vs. CEM (eight inner-relationship constraints for the hidden nodes are used); (b) EM vs. CEM (eight outer-relationship constraints for the hidden nodes are used); (c) CEM (eight outer-relationship constraints are used) vs. CEM (eight outer-relationship and eight inner-relationship constraints are used); (d) eight and 16 inner-relationship constraints are used, respectively; (e) eight and 16 outer-relationships are used, respectively; and (f) CEM (eight outer-relationship and eight inner-relationship constraints are used) vs. CEM (16 inner-relationship and 16 outer-relationship constraints are used). BN2 has the similar results.

Evaluation of performance. Two criteria are used to evaluate the performance of the learning algorithms. The first one is to compare the estimated CPT of each node to the ground-truth CPT using the Kullback–Leibler (KL) divergence. The smaller the KL-divergence is, the closer the estimated CPT to the true CPT. The second criterion is the negative log-likelihood per sample. It evaluates how well a learned BN fits the testing data as a whole. To compute that, we first generate 500 testing samples from each original BN. Each of the learned BNs is then evaluated on these samples to get the average negative log-likelihood. Since it is negative log-likelihood, the smaller the value is, the better the learned BN fits the data.

4.2. Learning results vs. type-I constraints

In this section, we compare learning performance when type-I constraints vary while type-II constraints are fixed. Specifically, both the set A and B include only hidden nodes. For type-I constraints, r varies from 0.2 to 0.4. For type-II constraints, only the inner-relationships for two parameters in the CPT of each hidden node are used as constraints.

Fig. 3 illustrates the results. In Charts (a) through (d), the x -coordinate denotes the node index, and the y -coordinate denotes the KL-divergence. The median of each bar is the mean, and the height of the bar is the standard deviation, which are obtained from 10 BN instances. Charts (a) and (b) compare EM to CEM when $r = 0.4$. We can see that CEM achieves better results in both mean and standard deviation of KL-divergence than EM for both BNs. In BN1, for the hidden nodes, the average mean decreases from 1.0687 (EM) to 0.2337 (CEM, $r = 0.4$); the average standard deviation decreases from 0.7659

(EM) to 0.2219 (CEM, $r = 0.4$). In BN2, for the hidden nodes, the average mean decreases from 0.6657 (EM) to 0.1163 (CEM, $r = 0.4$); the average standard deviation decreases from 0.5614 (EM) to 0.0969 (CEM, $r = 0.4$). Specifically, as shown in Charts (a) and (c), for the hidden node 18 in BN1, the KL-divergence decreases from around 2.1 (EM) to 0.2 (CEM, $r = 0.2$); for the hidden node 19 in BN1, the KL-divergence decreases from around 2.2 (EM) to 0.2 (CEM, $r = 0.2$).

Charts (c) and (d) compare CEM when r varies. As r decreases from 0.4 to 0.2, the performance of CEM is further improved, especially for the hidden nodes. The negative log-likelihood further confirms that CEM performs better than EM, as shown in Charts (e) and (f), where the x -coordinate indicates BN index and the y -coordinate indicates the negative log-likelihood. The negative log-likelihood from CEM is smaller than that from EM.

4.3. Learning results vs. type-II constraints

In the second scenario, we change type-II constraints while fix the type-I constraints (r is set as 0.4). We observe the learning results in the following cases: (1) varying the number of inner-relationship constraints; (2) varying the number of outer-relationship constraints; and (3) combining inner-relationship constraints and outer-relationship constraints.

Fig. 4 illustrates the results in the three cases. Charts (a) and (b) compare EM to CEM when eight inner-relationship constraints and eight outer-relationship constraints are used, respectively. Obviously, CEM always performs better than EM. For example, in Chart (a), the average mean for the hidden nodes decreases from 1.0687 (EM) to 0.2337 (CEM), the average standard deviation decreases

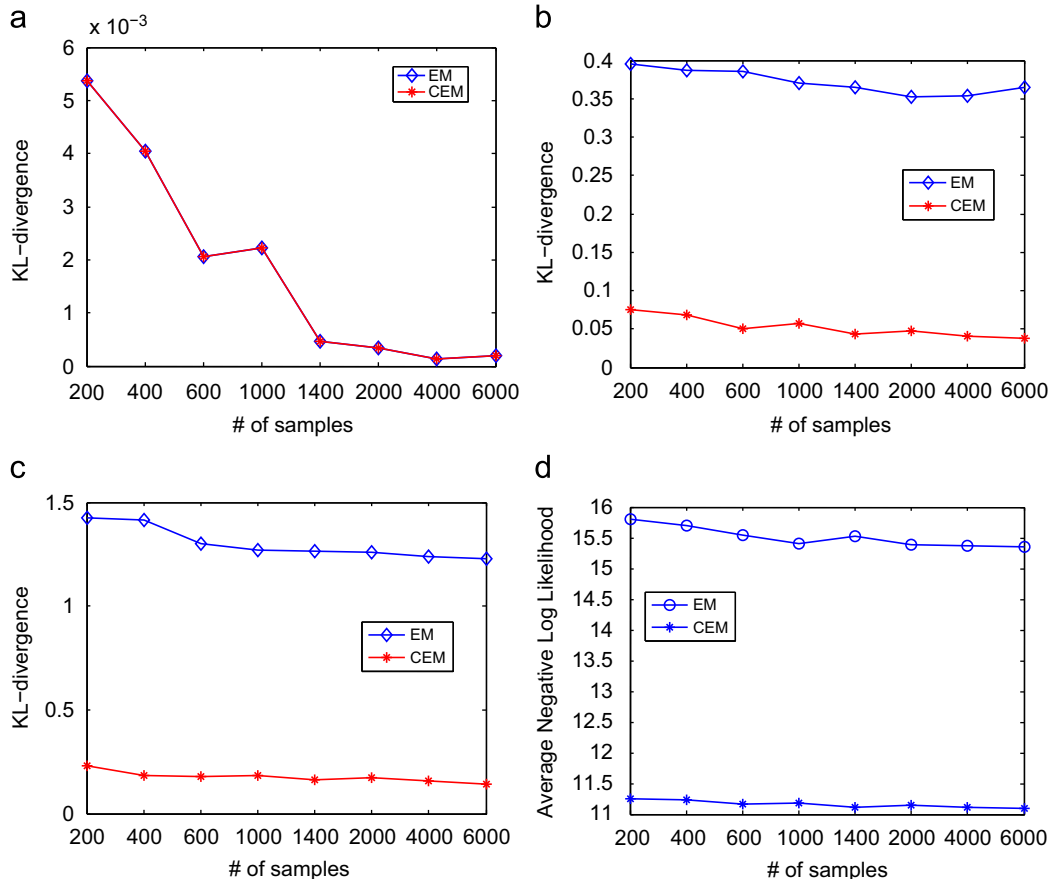


Fig. 5. Learning results vs. the number of training samples for BN1: (a) Average KL-divergence for the observable nodes whose parameters are independent from the hidden nodes (i.e., nodes 2, 3, 4, 9); (b) average KL-divergence for the other observable nodes; (c) average KL-divergence for the hidden nodes; and (d) negative log-likelihood. BN2 has the similar results.

from 0.7659 (EM) to 0.2219 (CEM); and in Chart (b), the average mean for the hidden nodes decreases from 1.0687 (EM) to 0.2214 (CEM), the average standard deviation decreases from 0.7659 (EM) to 0.2377 (CEM). Charts (c) through (f) show how the performance of CEM varies as different numbers of constraints are used. Chart (c) demonstrates that when both inner-relationship constraints and outer-relationship constraints are used, the performance is better than single-type constraints are used. The average mean for the hidden nodes decreases from 0.2214 to 0.1929, and the average standard deviation decreases from 0.2377 to 0.1763. Charts (d)–(f) show that the performance of CEM is improved slightly when we double the same types of constraints.

4.4. Learning results vs. training samples

We now demonstrate how the learning results vary with the number of training samples. In the experiments, we fix the constraints ($r=0.2$), and only eight inner-relationship constraints for the hidden nodes are used, but vary the number of training samples during learning. Fig. 5 demonstrates the results for BN1 only, since BN2 has the similar results. In order to observe how the training samples affect different types of nodes in BN1, we divide the nodes into three groups: group 1 includes the nodes (2, 3, 4, and 9) whose parameters are independent from the hidden nodes; group 2 includes the other observable nodes; and group 3 includes all the hidden nodes.

As shown in Chart (a) of Fig. 5, for the nodes in group 1, the KL-divergence decreases when the number of samples increases. And

the KL-divergence is very small in all the cases even when there are only 200 training samples. Both EM and CEM return the same results since the data are complete for all those nodes. In both Charts (b) and (c), the KL-divergence decreases slightly when the number of training samples increases, while the KL-divergence of CEM is much smaller than that of EM. Especially for the hidden nodes, even when the number of the training sample is 6000, the KL-divergence for the hidden nodes is only slightly smaller than the KL-divergence when the number of training sample is 200. This is because the information about the hidden nodes rarely varies as the total number of training samples increases. Therefore, in order to improve the learning results for the hidden nodes, domain knowledge is more important than training samples.

5. A case study

In this section, we apply our algorithm to a real-world application in computer vision: facial action unit (AU) recognition.

5.1. A Bayesian network for facial action unit modeling and recognition

In recent years, a variety of approaches are proposed to recognize facial expressions. Besides recognizing six basic facial expressions directly, techniques have also been developed to automatically recognize facial action units. According to the Facial Action Unit System (FACS) by Ekman [26], each AU is related to the contraction of a specific set of facial muscles. FACS has been demonstrated to be a powerful means for representing and characterizing a large number

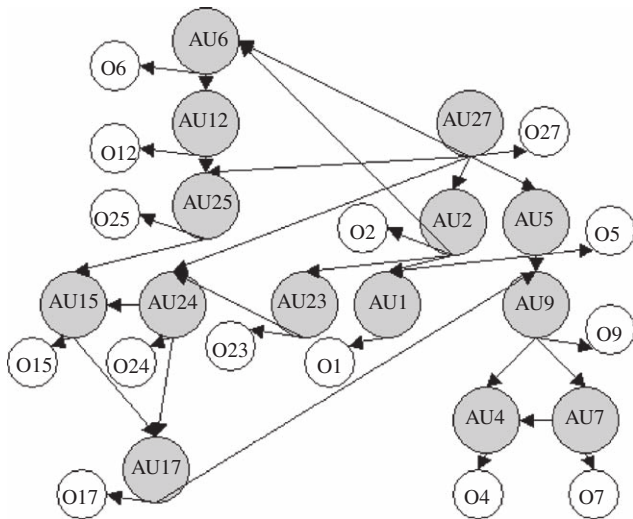


Fig. 6. A Bayesian network for AU modeling. We adapted it from [27]. The unshaded nodes are measurement nodes.

of facial expressions through the combination of only a small set of AUs.

Current techniques for recognizing AUs are mostly based on computer vision techniques. But due to the richness, ambiguity, and dynamic nature of facial actions, as well as the image uncertainty and individual difference, it is difficult to recognize each AU individually by using only computer vision techniques. Computer vision techniques should be combined with additional semantic information to achieve a more robust and consistent AU recognition. Fortunately, there are some inherent relationships among AUs as described in the FACS manual [26]. For example, the alternative rules provided in the FACS manual describe the mutual exclusive relationship among AUs. Furthermore, the FACS also includes the co-occurrence rules in their old version, which were “designed to make scoring more deterministic and conservative, and more reliable” [26].















Therefore, instead of recognizing each AU alone, a Bayesian network can be used to model the probabilistic relationships among different AUs, and given the BN model, AU recognition can be performed through a probabilistic inference [27]. Fig. 6 illustrates such a BN to model 14 AUs, and Table 3 summarizes a list of the 14 AUs and their meanings. They are adapted from [27]. Such a model is capable of representing the relationships among different AUs in a coherent and unified hierarchical structure, accounting for uncertainties in the AU recognition process with conditional dependence links, and providing principled inference and learning techniques to systematically combine the domain information and statistical information extracted from the data.

To incorporate the AU recognition results from a computer vision technique, in the BN model, each AU is connected with a measurement node (unshaded node), which encodes the measurement obtained from computer vision techniques. For AU measurement, we employ a technique similar to the one described in [28]. The output of the technique is a score for each AU, which is subsequently discredited to produce a value for a corresponding AU measurement node.

5.2. AU model parameter learning

Given the BN structure in Fig. 6 and the AU measurements, we then need parameterize the BN model before AU inference can complete. For this, we need training data. As defined before, a complete training sample requires the true AU label for each AU node

Table 3
A list of action units.

AU1  Inner brow raiser	AU2  Outer brow raiser	AU4  Brow Lowerer	AU5  Upper lid raiser	AU6  Cheek raiser
AU7  Lid tighten	AU9  Nose wrinkle	AU12  Lip corner puller	AU15  Lip corner depressor	AU17  Chin raiser
AU23  Lip tighten	AU24  Lip presser	AU25  Lips part	AU27  Mouth stretch	

and the measurement for each meresman node. However, manually labeling AUs is usually time consuming and difficult. Moreover, the labeling process is highly subjective, therefore prone to human errors. In addition, some AU events rarely happen in the collected data. Therefore, the training data could be incomplete, biased, or spare for certain AUs. We thus apply our algorithm to learn the BN parameters using only the AU measurements and some domain knowledge, without any AU labeling.

We first generate constraints. For type I constraints, domain experts are consulted to specify the approximate ranges for most of the parameters. Type II constraints are also constructed from domain specific knowledge. Specifically, for each measurement node, since the measurement accuracy for each AU varies, depending the computer vision technique used as well as on the difficulty of the AU, we can rank the measurements by their accuracy and then translate such a ranking into the outer-relationships between the corresponding measurement nodes. For example, the computer vision technique usually performs better in recognition of AU2 (outer brow raiser) than AU23 (lip tighten), hence we can get constraints like $p(O2=0|AU2=0) > p(O23=0|AU23=0)$, $p(O2=1|AU2=1) > p(O23=1|AU23=1)$, where 0 means an AU is absent, and 1 means an AU is present.

More type-II constraints can be obtained based on the properties of different AUs. For example, for AU6 (cheek raiser) and AU12 (lip corner puller), the probability of AU12 being absent if AU6 is absent, is smaller than the probability of AU6 being present if AU12 is present, i.e., $p(AU12=0|AU6=0) < p(AU12=1|AU6=1)$. For AU1 (inner brow raiser), the influence of AU2 (outer brow raiser) on AU1 is larger than the influence of AU5 (upper lid raiser) on AU1, i.e., $p(AU1=1|AU2=1, AU5=0) > p(AU1=1|AU2=0, AU5=1)$. Overall, we generate one type-I constraint for each parameter, and about 28 type-II constraints for all the parameters. Of course, the number of constraints depends on the application and the domain knowledge available for the application.

5.3. AU recognition results

We use 8000 images collected from Cohan and Kanade’s DFAT-504 database [29], where 80% are used for training and 20% data are used for testing. We first use MLE to learn parameters from the complete data, which consists both the true AU labels and the AU measurements. Then, we use the EM and CEM algorithms to learn parameters from the incomplete data, which only includes the AU measurements.

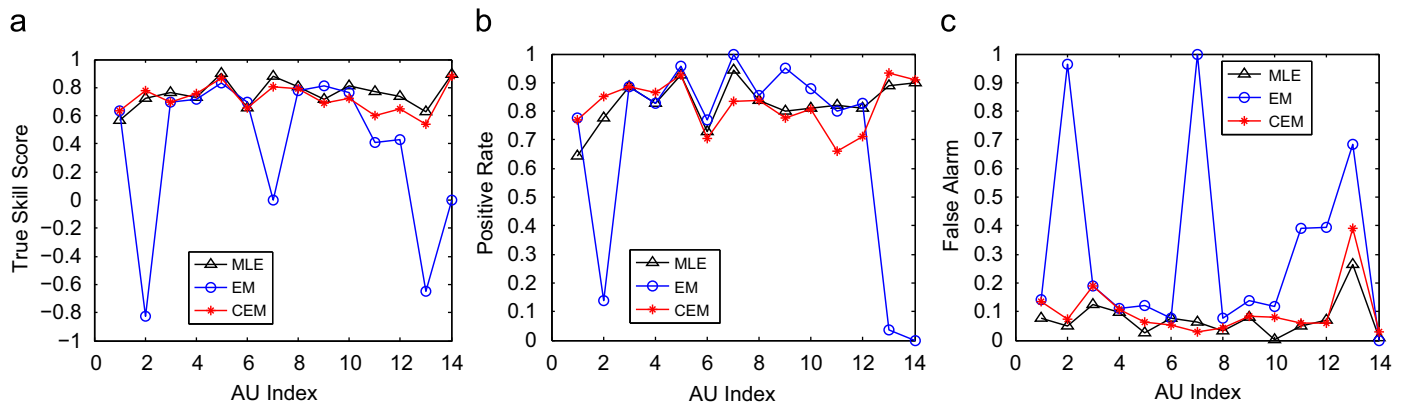


Fig. 7. Comparison of average AU recognition results using the BNs learned from MLE, EM, and CEM, respectively: (a) true skill score; (b) positive rate; and (c) false alarm. MLE uses complete training data, while EM and CEM use incomplete data that only include AU measurements.

Fig. 7 compares the AU recognition results with BNs learned from MLE, EM, and CEM in terms of true skill score (the difference between the positive rate and the false alarm), positive rate, and false alarm. CEM performs similarly to MLE (based on complete data), but much better than EM. The average true skill score is only 0.37 for EM, 0.72 for CEM, and 0.76 for MLE. The positive rate increases from 0.69 (EM) to 0.82 (CEM), and the false alarm decreases from 0.32 (EM) to 0.1 (CEM). For EM, some AUs totally fail, such as AU2, AU7, and AU23. But CEM has a fair performance for all the AUs even it is learned from the unlabeled data. This again shows the importance of domain knowledge. CEM is able to fully utilize the domain knowledge for automatic parameter learning. Compared to MLE that is based on the labeled data, the CEM has comparable performance but without using any labeled AUs. This is indeed very encouraging. This may represent a significant step forward in machine learning in general and BN learning in particular.

6. Conclusion and future work

When a large amount of data are missing, or when multiple hidden nodes exist, learning parameters in Bayesian networks becomes extremely difficult. The learning algorithms are required to operate in a high-dimensional search space and could easily get trapped among copious local maxima. We thus present a constrained EM algorithm to learn Bayesian network parameters when a large amount of data are missing in the training data. The algorithm fully utilizes certain qualitative domain knowledge to regularize the otherwise ill-posed problem, limit the search space, and avoid local maxima. Compared with the quantitative domain knowledge such as prior probability distribution typically used by the existing methods, the qualitative domain knowledge is local (only concerned with some parameters), easy to specify, and does not need strong assumption.

For many computer vision and pattern recognition problem, data is often hard to acquire and the model becomes increasingly complex. It, therefore, becomes increasingly important to incorporate human knowledge into the otherwise ill-posed learning process. Our method can solicit simple yet effective qualitative constraints from human experts, and then systematically incorporate them into the learning process. The improvement in learning performance is significant. Both the experiments from the synthetic data and real data for facial action recognition demonstrate that our algorithm improves the accuracy of the learned parameters significantly over the traditional EM algorithm.

The domain knowledge in the current learning algorithm was formalized by two simple constraints: a range of a parameter, and

relative relationships between different parameters. Although they are very useful, it is possible to introduce more types of constraints into learning, such as the relationships between the sum of several parameters, parameter sharing, etc. More constraints will help further reduce the search space, although they may not be easy for domain experts to specify.

Furthermore, we assumed model structures are known and thus only focused on learning model parameters. However, in many applications, the structure could be unknown, or it is too difficult for domain experts to manually construct a complete structure. Learning the BN structure is therefore also necessary. Most current approaches to BN structure learning assume generic prior probabilities on graph structures, typically encoding a sparseness bias but otherwise expecting no regularities in the learned structures. We believe that the domain knowledge about model parameters can also help in learning the model structure.

References

- [1] E. Delage, H. Lee, A. Ng, A dynamic Bayesian network model for autonomous 3d reconstruction from a single indoor image, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2006.
- [2] J.M. Pena, J. Bjoerkegren, J. Tegner, Growing Bayesian network models of gene networks from seed genes, *Bioinformatics* (2005) 224–229.
- [3] L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, Bayesian networks and information retrieval: an introduction to the special issue, *Information Processing and Management* (2004) 727–733.
- [4] Y. Zhang, Q. Ji, Active and dynamic information fusion for facial expression understanding from image sequence, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (5) (2005) 699–714.
- [5] N. Friedman, M. Goldszmidt, D. Heckerman, S. Russell, Where is the impact of Bayesian networks in learning? in: *International Joint Conference on Artificial Intelligence*, 1997.
- [6] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *The Royal Statistical Society Series B* 39 (1977) 1–38.
- [7] S. Geman, D. Geman, Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (1984) 721–741.
- [8] M. Jaeger, The AI&M procedure for learning from incomplete data, in: *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 2006, pp. 225–232.
- [9] M. Ramoni, P. Sebastiani, Robust learning with missing data, *Machine Learning* 45 (2) (2001) 147–170.
- [10] R.G. Cowell, Parameter learning from incomplete data using maximum entropy I: principles, *Statistical Research Report*, vol. 21, 1999.
- [11] R.G. Cowell, Parameter learning from incomplete data using maximum entropy II: application to Bayesian networks, *Statistical Research Report*, vol. 21, 1999.
- [12] G. Elidan, N. Friedman, The information bottleneck EM algorithm, in: *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 2003, pp. 200–209.
- [13] G. Elidan, M. Ninio, N. Friedman, D. Schuurmans, Data perturbation for escaping local maxima in learning, in: *Proceedings of the 18th National Conference on Artificial Intelligence*, 2002, pp. 132–139.

- [14] B. Thiesson, Accelerated quantification of Bayesian networks with incomplete data, in: Proceedings of the First International Conference on Knowledge Discovery and Data Mining, 1995, pp. 306–311.
- [15] E. Bauer, D. Koller, Y. Singer, Update rules for parameter estimation in Bayesian networks, in: Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence, 1997, pp. 3–13.
- [16] W.L. Buntine, Operations for learning with graphical models, *Artificial Intelligence Research* 2 (1994) 159–225.
- [17] S.T. Lauritzen, The EM algorithm for graphical association models with missing data, *Computational Statistics and Data Analysis* 19 (1995) 191–201.
- [18] R.S. Niculescu, T.M. Mitchell, R.B. Rao, A theoretical framework for learning Bayesian networks with parameter inequality constraints, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence, 2007.
- [19] M.J. Druzdzel, L.C. van der Gaag, Elicitation of probabilities for belief networks: combining qualitative and quantitative information, in: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 1995, pp. 141–148.
- [20] F. Wittig, A. Jameson, Exploiting qualitative knowledge in the learning of conditional probabilities of Bayesian networks, in: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, 2000, pp. 644–652.
- [21] E.E. Altendorf, A.C. Restificar, T.G. Dietterich, Learning from sparse data by exploiting monotonicity constraints, in: Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence, 2005, pp. 18–26.
- [22] A. Feelders, L. van der Gaag, Learning Bayesian network parameters with prior knowledge about context-specific qualitative influences, in: Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence, 2005, pp. 193–20.
- [23] S. Russell, J. Binder, D. Koller, K. Kanazawa, Local learning in probabilistic networks with hidden variables, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 1146–1152.
- [24] J. Binder, D. Koller, S. Russell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, *Machine Learning* (1997) 213–244.
- [25] A. Feelders, L. van der Gaag, Learning Bayesian network parameters under order constraints, *International Journal of Approximate Reasoning* 42 (2006) 37–53.
- [26] P. Ekman, W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*, Consulting Psychologists Press, Palo Alto, CA, 1978.
- [27] Y. Tong, W. Liao, Q. Ji, Inferring facial action units with causal relations, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2006.
- [28] M.S. Bartlett, G. Littlewort, M. Frank, C. Lainscek, I. Fasel, J. Movellan, Recognizing facial expression: machine learning and application to spontaneous behavior, *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition* 2 (2005) 568–573.
- [29] T. Kanade, J.F. Cohn, Y. Tian, Comprehensive database for facial expression analysis, in: Proceedings of FG00, 2000.

About the Author—WENHUI LIAO received the PhD degree from the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, New York, in 2006. Her areas of research include probabilistic graphical models, information fusion, computer vision, and natural language processing. She is currently a research scientist at R&D of Thomson-Reuters Corporation.

About the Author—QIANG JI received the PhD degree in electrical engineering from the University of Washington in 1998. He is currently an associate professor in the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute (RPI), Troy, New York. Prior to joining RPI in 2001, he was an assistant professor in the Department of Computer Science, University of Nevada, Reno. He also held research and visiting positions with Carnegie Mellon University, Western Research, and the US Air Force Research Laboratory. His research interests include computer vision, probabilistic reasoning with Bayesian networks for decision making and information fusion, human computer interaction, pattern recognition, and robotics. He has published more than 100 papers in peer-reviewed journals and conferences. His research has been funded by local and federal government agencies including the US National Science Foundation (NSF), the US National Institute of Health (NIH), the US Air Force Office of Scientific Research (AFOSR), the US Office of Naval Research (ONR), DARPA, and the US Army Research Office (ARO) and by private companies including Boeing and Honda. He is a senior member of the IEEE.

Strategy Selection in Influence Diagrams using Imprecise Probabilities

Cassio P. de Campos

Electrical, Computer and Systems Eng. Dept.
Rensselaer Polytechnic Institute
Troy, NY, USA
decamc@rpi.edu

Qiang Ji

Electrical, Computer and Systems Eng. Dept.
Rensselaer Polytechnic Institute
Troy, NY, USA
jiq@rpi.edu

Abstract

This paper describes a new algorithm to solve the decision making problem in Influence Diagrams based on algorithms for credal networks. Decision nodes are associated to imprecise probability distributions and a reformulation is introduced that finds the global maximum strategy with respect to the expected utility. We work with Limited Memory Influence Diagrams, which generalize most Influence Diagram proposals and handle simultaneous decisions. Besides the global optimum method, we explore an anytime approximate solution with a guaranteed maximum error and show that imprecise probabilities are handled in a straightforward way. Complexity issues and experiments with random diagrams and an effects-based military planning problem are discussed.

1 INTRODUCTION

An influence diagram is a graphical model for decision making under uncertainty [13]. It is composed by a directed graph where utility nodes are associated to profits and costs of actions, chance nodes represent uncertainties and dependencies in the domain and decision nodes represent actions to be taken. Given an influence diagram, a strategy defines which decision to take at each node, given the information available at that moment. Each strategy has a corresponding expected utility. One of the most important problems in influence diagrams is *strategy selection*, where we need to find the strategy with maximum expected utility. A simple approach is to evaluate each possible strategy and compare their expected utilities. However, the number of strategies grows exponentially in the number of decision to be taken.

In this paper, we propose a new idea to find the best

strategy based on a reformulation of the problem as an inference in a credal network [4]. We show through experiments that this approach can handle small and medium diagrams exactly, and provides an anytime approximation in case we stop the process early. Our idea works with a very general class of influence diagrams, named *Limited Memory Influence Diagrams* (LIMIDs) [15]. *Limited Memory* means that the assumption of *no-forgetting* usually employed in Influence Diagrams (that is, values of observed variables and decisions that have been taken are remembered at all later times) is relaxed. This class of diagrams is interesting because most other influence diagram proposals can be efficiently converted into LIMIDs.

To solve strategy selection, many approaches work on special cases of influence diagrams, exploiting their characteristics to improve performance. In many cases, it is assumed that there is an ordering on which the decisions are to be taken and the no-forgetting rule, so as previous decisions are assumed to be known in the moment of the current decision [14, 18, 19, 20, 21]. The ordering of decision nodes is exploited to evaluate the optimal strategy. There are also proposals in the class of simultaneous influence diagrams, where decisions are assumed to have no antecedents. This assumption reduces the number of possible strategies and allows for factorization ideas [22]. LIMIDs do not have assumptions about no-forgetting and ordering for decisions, even though it is possible to convert diagrams that have such assumptions into LIMIDs.

In order to test our method, we generate a data set of random influence diagrams. Empirical results indicate that the accuracy of our method is better than other approaches'. We also apply our idea to solve an Effects-based operations (EBO) military planning. The EBO approach seeks for a campaign objective by considering direct, indirect and cascading effects of military, diplomatic, psychological and economic actions [6, 11]. We use an influence diagram to model an EBO hypothetical problem.

Section 2 introduces our notation for influence diagrams and the problem of strategy selection. Section 3 describes the framework of credal networks and the inference problem on such networks. Section 4 presents how we solve strategy selection through a reformulation of the problem as an inference in credal networks. Section 5 presents some experiments, including the EBO military planning problem, and finally Section 6 concludes the paper and indicates future work.

2 INFLUENCE DIAGRAMS

A Limited Memory Influence Diagram \mathcal{I} is composed by a directed acyclic graph (\mathcal{V}, E) where nodes are partitioned in three types: chance, decision and utility nodes. Let \mathcal{C} , \mathcal{D} and \mathcal{U} be the set of chance, decision and utility nodes, respectively, and let $\mathcal{X} = \mathcal{C} \cup \mathcal{D}$. Links of E characterize dependencies among nodes. Explicitly, links toward a chance node indicate probabilistic dependence of the node on its parents; links toward a decision node indicate which information is available to take such decision, and links toward utility nodes represent that an utility for those parents is to be considered (utility nodes may not have children). Associated to each node, there are some parameters:

1. A *chance node* has an associated categorical random variable C with finite domain Ω_C and conditional probability distributions $p(C|\pi_j(C))$, for each configuration $\pi_j(C)$ of its parents $\pi(C)$ in the graph. j is used to indicate a configuration of the parents of C , that is, $\pi_j(C) \in \Omega_{\pi(C)}$, where the notation $\Omega_{\mathcal{V}'} = \times_{V \in \mathcal{V}'} \Omega_V$, for any $\mathcal{V}' \subseteq \mathcal{V}$.
2. A *decision node* D is associated to a finite set of mutually exclusive alternatives Ω_D . Parents of D describe the information that is available at the moment on which decision D has to be taken.
3. An *utility node* U is associated to a rational function $f_U : \Omega_{\pi(U)} \rightarrow \mathcal{Q}$. The value corresponding to a parent configuration is the profit (cost is viewed as negative profit) of such parent configuration. Utility nodes have no children.

A simple example is depicted in Figure 1. Decision nodes are represented by rectangles, chance nodes by ellipses and utility nodes by diamonds. *do_ground_attack* has an associated cost, which is depicted by the corresponding utility node. The same is modeled for *bomb_bridge*. The goal is to achieve *territory_occupation*, which also has an utility (the profit of the goal). *ground_attack* and *bridge_condition* represent the uncertain outcomes of the corresponding actions. Note that there is no known ordering on which

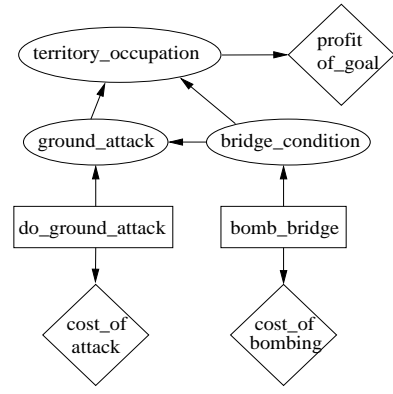


Figure 1: Simple Influence Diagram example.

decisions must be taken. Although decision nodes have no parents in this example, there is no such restriction.

A *policy* δ_D for the decision node D is a function $\delta_D : \Omega_{D \cup \pi(D)} \rightarrow [0, 1]$ defined for each alternative of D and each configuration of $\pi(D)$ such that, for each $\pi_j(D) \in \Omega_{\pi(D)}$ we have $\sum_{d \in \Omega_D} \delta_D(d, \pi_j(D)) = 1$. A *pure policy* is a policy such that its image is integer ($\delta_D : \Omega_{D \cup \pi(D)} \rightarrow \{0, 1\}$), and thus specifies with certainty which action (alternative of D) is taken for each parent configuration (in a pure policy, only one $\delta_D(d, \pi_j(D))$ for each $\pi_j(D)$ will be non-zero as they sum 1). A *strategy* Δ is a set of policies $\{\delta_D : D \in \mathcal{D}\}$, one for each decision node of the diagram. A *pure strategy* is composed only by pure policies.

The expected utility $EU(\Delta)$ of a strategy Δ is evaluated through the following equation:

$$\sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} \left(\prod_C p(x_C | \pi_j(C)) \prod_D \delta_D(x_D) \sum_U f_U(\pi_{j'}(U)) \right), \quad (1)$$

where x_C , $\pi_j(C)$, x_D and $\pi_{j'}(U)$ are respectively the projections of \mathbf{x} in Ω_C , $\Omega_{\pi(C)}$, $\Omega_{D \cup \pi(D)}$ and $\Omega_{\pi(U)}$. This equation means that, given a strategy, its expected utility is the sum of the utility values weighted by the probability of each diagram configuration (for all configurations). The maximum expected utility is obtained over all possible strategies:

$$MEU = \max_{\Delta} EU(\Delta).$$

The problem of *strategy selection* is to obtain the strategy that maximizes its expected utility, that is, $\text{argmax} \max_{\Delta} EU(\Delta)$.

3 CREDAL NETWORKS

We need some concepts of credal networks before presenting the reformulation to solve strategy selection. A convex set of probability distributions is called a

credal set [4]. A credal set for X is denoted by $K(X)$; we assume that every random variable is categorical and that every credal set has a finite number of vertices. Given a credal set $K(X)$ and an event A , the *upper* and *lower* probability of A are respectively $\max_{p(X) \in K(X)} p(A)$ and $\min_{p(X) \in K(X)} p(A)$. A conditional credal set is a set of conditional distributions, obtained by applying Bayes rule to each distribution in a credal set of joint distributions.

A (separately specified) *credal network* $N = (G, \mathbb{X}, \mathbb{K})$ is composed by a directed acyclic graph $G = (V, E)$ where each node of V is associated with a random variable $X_i \in \mathbb{X}$ and with a collection of conditional credal sets $K(X_i | \pi(X_i)) \in \mathbb{K}$, where $\pi(X_i)$ denotes the parents of X_i in the graph. Note that we have a conditional credal set related to X_i for each configuration $\pi_j(X_i) \in \Omega_{\pi(X_i)}$. A root node is associated with a single marginal credal set. We take that in a credal network every random variable is independent of its non-descendants non-parents given its parents; this is the *Markov condition* on the network. In this paper we adopt the concept of *strong independence*¹: two random variables X_i and X_j are strongly independent when every extreme point of $K(X_i, X_j)$ satisfies standard stochastic independence of X_i and X_j (that is, $p(X_i | X_j) = p(X_i)$ and $p(X_j | X_i) = p(X_j)$) [4]. Strong independence is the most commonly adopted concept of independence for credal sets, probably due to its connection with standard stochastic independence.

Given a credal network, its *extension* is any joint credal set that satisfies all constraints encoded in the network. The *strong extension* \mathcal{K} of a credal network is the largest joint credal set such that every variable is strongly independent of its non-descendants non-parents given its parents. The strong extension of a credal network is the joint credal set that contains every possible combination of vertices for all credal sets in the network [5]; that is, each vertex of a strong extension factorizes as follows:

$$p(X_1, \dots, X_n) = \prod_i p(X_i | \pi(X_i)). \quad (2)$$

Thus, a credal network can be viewed as a representation for a set of Bayesian networks with distinct parameters but sharing the same graph.

3.1 INFERENCE

A *marginal inference* in a credal network is the computation of upper (or lower) probabilities in an extension of the network. If X_q is a *query* variable, then a marginal inference is the computation of tight bounds

¹We note that other concepts of independence are found in the literature [3, 10].

for $p(x_q)$ for one or more categories x_q of X_q . For inferences in strong extensions, it is known that distributions that maximize $p(x_q)$ belong to the set of vertices of the extension [12]. So, an inference can be produced by combinatorial optimization, as we must find a vertex for each local credal set $K(X_i | \pi(X_i))$ so that Expression (2) leads to a maximum of $p(x_q)$. In general, inference offers tremendous computational challenges, and exact inference algorithms based on enumeration of all potential vertices face serious difficulties [4].

A different way to solve the problem is to recognize that an upper (or lower) value for $p(x_q)$ may be obtained by the optimization of a multilinear polynomial over probability values, subject to constraints. This idea is discussed in the literature and different methods to reformulate the inference problem were proposed [7, 9]. Empirical results suggest that this is the most effective way for exact inferences. In the next section, we describe an idea based on bilinear programming [9] to perform inferences in credal networks and show how it can be employed to solve the strategy selection problem of influence diagrams.

4 STRATEGY SELECTION AS A CREDAL NET INFERENCE

Suppose we want to find the strategy Δ_{opt} that maximizes the expected utility in an influence diagram \mathcal{I} , that is, $\Delta_{opt} = \operatorname{argmax} MEU$. Let \underline{f} and \bar{f} be the minimum and maximum utility values specified in the diagram for all possible utility nodes and parent configurations, that is,

$$\underline{f} = \min_{U, \pi_j(U)} f_U(\pi_j(U)), \quad \bar{f} = \max_{U, \pi_j(U)} f_U(\pi_j(U)).$$

We create an identical influence diagram \mathcal{I}' except that the utility function f'_U (for each node U) is defined as

$$\forall \pi_j(U) \quad f'_U(\pi_j(U)) = \frac{f_U(\pi_j(U)) - \underline{f}}{\bar{f} - \underline{f}}.$$

The denominator is positive because $\underline{f} < \bar{f}$ (if $\underline{f} = \bar{f}$, then the influence diagram is trivial as all utility values are equal). We note that this transformation is similar to that proposed by Cooper [2]. It is not hard to see that $\operatorname{argmax} MEU = \operatorname{argmax} MEU'$ (just take the terms out of summations in Equation (1)), and

$$\max_{\Delta} EU'(\Delta) = \frac{\max_{\Delta} EU(\Delta) - |\mathcal{U}| \underline{f}}{\bar{f} - \underline{f}}.$$

This implies that strategy selection in \mathcal{I} is the same as strategy selection in \mathcal{I}' . Now, we translate the selection problem of \mathcal{I}' to a credal network inference. Suppose we define a credal network with a similar graph as \mathcal{I}' such that:

- Chance nodes are directly translated as nodes of the credal network (parents are the same as in \mathcal{I}').
- Utility nodes are translated to binary random nodes. Let U be an utility node with function f_U . In the credal network, U becomes a binary node (with the same parents as before) and categories u and $\neg u$ such that: $p(u|\pi_j(U)) = f_U(\pi_j(U))$ and $p(\neg u|\pi_j(U)) = 1 - p(u|\pi_j(U))$ [2].
- Decision nodes are translated to probabilistic nodes with imprecise distributions such that policies become probability distributions (in fact, according to our definition of policy, they are already greater than zero and sum 1). Thus, $p(d|\pi_j(D)) = \delta_D(d, \pi_j(D))$ for all d and $\pi_j(D)$. Note that $p(D|\pi_j(D))$, for each $\pi_j(D)$, is a distribution with unknown probability values (this interpretation of decision nodes as imprecise probability nodes is discussed by Antonucci and Zafalon, see e.g. [1]).

Using this credal network formulation, the expected utility of a strategy Δ can be written as

$$EU'(\Delta) = \sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} \left(\prod_X p_{\Delta}(x|\pi_j(X)) \sum_U p(u|\pi_{j'}(U)) \right),$$

where x , $\pi_j(X)$ and $\pi_{j'}(U)$ are projections of \mathbf{x} into the corresponding domains, X ranges on all nodes corresponding to chance and decision nodes of the influence diagram, and p_{Δ} represents the distribution induced by the strategy Δ , that is, when the strategy is chosen, p_{Δ} is a known probability distribution.

With some simple manipulations, we have:

$$\begin{aligned} EU'(\Delta) &= \sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} \left(p_{\Delta}(\mathbf{x}) \sum_U p(u|\pi_{j'}(U)) \right), \\ EU'(\Delta) &= \sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} \left(\sum_U p(u|\pi_{j'}(U)) p_{\Delta}(\mathbf{x}) \right), \\ EU'(\Delta) &= \sum_U \sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} p_{\Delta}(u, \mathbf{x}) = \sum_U p_{\Delta}(u), \end{aligned}$$

and then

$$MEU' = \max_{\Delta} \sum_U p_{\Delta}(u) = \max_{p \in \mathcal{K}} \sum_U p(u),$$

where $p \in \mathcal{K}$ means that we select a distribution p in the extension of the credal network. In fact the only places p may vary are related to the imprecise probabilities of the former decision nodes. When we select p , we get a precise distribution that has a corresponding strategy Δ . So, we have a credal network and need to find a distribution p that maximizes the sum of marginal probabilities of the U nodes.

4.1 INFERENCE AS AN OPTIMIZATION PROBLEM

The sum of marginal inferences in the credal network can be formulated as a multilinear programming problem. The goal is to maximize the expression

$$\sum_U p(u) = \sum_U \sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} \left(p(u|\pi_{j'}(U)) \prod_X p(x|\pi_j(X)) \right), \quad (3)$$

where x , $\pi_{j'}(U)$ and $\pi_j(X)$ are the projections of \mathbf{x} in the corresponding domains, and where some distributions $p(X|\pi_j(X))$ are precisely known and others are imprecise. In this formulation we must deal with a large number of multilinear terms. To avoid them, we briefly describe the bilinear transformation procedure proposed by de Campos and Cozman [9] to replace the large Expression (3) by simple bilinear expressions. We refer to [9] for additional details.

The idea is based on a *precedence ordering* of the network variables, which is an ordering where all ancestors of a given variable in the network's graph appear before it in the ordering. The bilinear transformation algorithm processes the network variables top-down: at each step some constraints are generated that define the relationship between the query and the current variable being processed. A variable may be processed only if all its ancestors have already been processed. The active nodes at each step form a path-decomposition of the network's graph.

To better explain the method, we take the example of Figure 1. For simplicity, assume that variables are binary² (with categories b and $\neg b$) renamed as follows: *do_ground_attack* is D_1 , *bomb_bridge* is D_2 , *cost_of_attack* is U_1 , *cost_of_bombing* is U_2 , *ground_attack* is C_1 , *bridge_condition* is C_2 , *territory_occupation* is C_3 , and finally *profit_of_goal* is U_3 .

After the translation of the utility functions into probability distributions and the replacement of decision nodes by nodes with imprecise probabilities (as previously described), we have a credal network and need to maximize the sum of the marginal probabilities of the U nodes. In fact this is an extension of the standard query in a credal network, because we have a summation instead of a single probability to maximize. So the objective function is $\max p(u_1) + p(u_2) + p(u_3)$ (there are three utility nodes in the example) subject to constraints that define each marginal probability $p(u_1)$, $p(u_2)$ and $p(u_3)$. To create these constraints, we run a symbolic inference based on the precedence ordering for each of the marginal probabilities. The constraints for $p(u_1)$ and $p(u_2)$ are very

²The method works on non-binary variables as well. The assumption is made here for ease of expose.

simple: $p(u_1) = p(u_1|d_1)p(d_1) + p(u_1|\neg d_1)p(\neg d_1)$ and $p(u_2) = p(u_2|d_2)p(d_2) + p(u_2|\neg d_2)p(\neg d_2)$, because they only depend on one other variable. Note that $p(d_1)$, $p(\neg d_1)$, $p(d_2)$, and $p(\neg d_2)$ that appear in these constraints are unknown and thus become optimization variables in the bilinear problem.

To write the constraints for $p(u_3)$, we need to choose a precedence ordering. We will use the ordering $D_2, C_2, D_1, C_1, C_3, U_3$ (variables U_1 and U_2 do not appear in the order as they are not relevant to evaluate the marginal $p(u_3)$). Hence, the first variable to be processed is D_2 . We write a constraint that relates the query u_3 and probabilities $p(D_2)$ (which are defined in the network specification):

$$p(u_3) = \sum_{d \in \{d_2, \neg d_2\}} p(d) \cdot p(u_3|d).$$

D_2 now appears in the conditional part of $p(u_3|d)$, which may be viewed as an artificial term in the optimization, as it does not appear in the network. Because of that, we must create constraints to define $p(u_3|d)$ in terms of network parameters (for all categories $d \in D_2$). According to our chosen ordering, the current variable to be processed is C_2 . Thus,

$$\begin{aligned} p(u_3|d_2) &= \sum_{c \in \{c_2, \neg c_2\}} p(c|d_2) \cdot p(u_3|c), \\ p(u_3|\neg d_2) &= \sum_{c \in \{c_2, \neg c_2\}} p(c|\neg d_2) \cdot p(u_3|c). \end{aligned}$$

Note that $p(u_3|c) = p(u_3|c, d)$ (for any d), so we use the simpler. At this stage, our query is conditioned on C_2 . Following the same idea, we process D_1 , obtaining

$$\begin{aligned} p(u_3|c_2) &= \sum_{d \in \{d_1, \neg d_1\}} p(d) \cdot p(u_3|c_2, d), \\ p(u_3|\neg c_2) &= \sum_{d \in \{d_1, \neg d_1\}} p(d) \cdot p(u_3|\neg c_2, d). \end{aligned}$$

Now the current variable to be treated is C_1 , and our query is conditioned on C_2, D_1 , that is, we must define how to evaluate $p(u_3|C_2, D_1)$ for all configurations. Thus, for all $c \in \{c_2, \neg c_2\}$ and $d \in \{d_1, \neg d_1\}$:

$$p(u_3|c, d) = \sum_{c' \in \{c_1, \neg c_1\}} p(c'|c, d) \cdot p(u_3|c, c').$$

At this moment, u_3 is conditioned on C_1, C_2 in the artificial term $p(u_3|c, c')$ (D_1 is not present in the artificial term as C_1, C_2 separate u_3 from D_1). Now we process C_3 : for all $c' \in \{c_1, \neg c_1\}$ and $c \in \{c_2, \neg c_2\}$

$$p(u_3|c, c') = \sum_{c'' \in \{c_3, \neg c_3\}} p(c''|c, c') \cdot p(u_3|c'').$$

Note that, as $p(u_3|c'')$ is specified in the network, we can stop. All artificial terms are related (through constraints) to parameters of the network. Besides all these constraints, we also include simplex constraints to ensure that probabilities sum 1.

Hence, we have a collection of linear and bilinear constraints on which non-linear programming can be employed [7]. It is also possible to use linear integer programming [9]. The steps to achieve a linear integer programming formulation are simple, because the only non-linear terms of the problem have the format $b \cdot t$, where $b \in \{0, 1\}$ and $t \in [0, 1]$. b is an unknown probability value of the credal network (which is zero or one because the solution we look for lies on extreme points of credal sets [12]) and t is a constant or an artificial term created in the procedure just described. To *linearize* the problem, $b \cdot t$ is replaced by an additional artificial optimization variable y and the following constraints are inserted: $0 \leq y \leq b$ and $t - 1 + b \leq y \leq t$. After replacing all non-linear terms using this idea, the problem becomes a linear integer programming problem, where a solution is also a solution for the strategy selection in the initial influence diagram.

We emphasize that, as we are translating the strategy selection problem into a credal network inference, it is straightforward to use imprecise probabilities in the chance nodes of the influence diagram. Intervals or sets of probabilities may be used. The translation works in the same way, but the generated problem will have more imprecise probabilities to optimize.

The following theorem shows that, when reformulating the strategy selection problem as a modified credal network inference, we are not making use of “more effort” than necessary, that is, strategy selection has the same complexity as inference in credal networks.

Theorem 1 *Let \mathcal{I} be a LIMID and k a rational. Deciding whether there is a strategy Δ such that MEU is greater than k is NP-Complete when \mathcal{I} has bounded induced width,³ and NP^{PP} -Complete in general.*

Proof sketch: Pertinence for the bounded induced width case is achieved because (given a strategy) we can compute MEU and verify if it is greater than k in polynomial time (using the reformulation and the sum of marginal queries, each marginal query takes polynomial time in a bounded induced width Bayesian network); in the general case, we can perform this verification using a PP oracle. Hardness for the bounded induced width case is obtained with the same reduc-

³The maximum clique and the maximum degree in the moral graph are bounded by a logarithmic function in the size of the input needed to specify the problem, which for instance includes polytrees.

tion as in [8] from the MAXSAT problem (replacing the credal nodes with decision nodes and introducing a single utility node). In the general case, the same reduction as in [17] from E-MAJSAT can be used (MAP nodes are replaced by decision nodes). \square

5 EXPERIMENTS

We conduct two experiments with the procedure. First, we use random generated influence diagrams to compare the solutions obtained by our procedure (which we call CR for *credal reformulation*) against the *Single Policy Updating* (SPU) of Lauritzen and Nilsson [15]. Later we work with a practical EBO military planning problem and compare the method against the factorization of Zhang and Ji [22].⁴

Concerning random influence diagrams, we have generated a data set based on the total number of nodes and the number of decision nodes. The configurations chosen are presented in the first two columns of Table 1. We have from 10 to 120 nodes, where 3 to 35 are decision nodes. The number of utility nodes is chosen equal to the number of decision nodes. Each line in Table 1 contains the average result for 30 random generated diagrams within that configuration. The third column of the table shows the approximate average number of distinct strategies in the diagrams that would need to be evaluated by a brute force method.

The three columns of the CR method show the time spent to solve the problem, the number of nodes evaluated in the branch-and-bound tree of the optimization procedure (which is significantly smaller than the total number of strategies in brute force) and the maximum error of the solution (all numbers are averages). After the reformulation, the CPLEX solver [16] is used, which includes a heuristic search before starting the branch-and-bound procedure. The evaluations of this heuristic search are not counted in the fifth column of Table 1. Note that the first five rows are separated from the last three because they strongly differ on the size of the search space (exact solutions were found only for the former). The maximum error of each solution is obtained straightforward from the relaxation of the linear integer problem. The last two columns of Table 1 show the time and maximum error of the SPU approximate procedure. Although very fast, the SPU procedure has worse accuracy than the “approximate” CR (solution was approximate in last three rows because we have imposed a time-limit of ten minutes for each run). Furthermore, SPU does not provide an upper bound for the best possible expected utility, as obtained by CR. Still, a possible improvement is to use

⁴The factorization idea only works on simultaneous influence diagrams, so it was not used in the other test cases.

SPU to provide an initial guess to the optimization.

5.1 EBO MILITARY PLANNING

In this section we describe the performance of our method in an hypothetical Effects-based Operations planning problem [11]. An influence diagram similar to the model described by Zhang and Ji [22] is employed. Its graph is shown in Figure 2. The goal is to win a war, which is represented by the *Hypothesis* node (on top of Figure 2). Just below there are the subgoals *Air_superiority*, *Territory_occupation*, and *Commander_surrender*, which are directly related to the main goal. There are eleven decision nodes (represented by rectangles): *destroy_C2* (C2 stands for *Command and Control*), *destroy_Radars*, *destroy_Communications*, *launch_air_strike*, *destroy_RD*, *destroy_storage*, *destroy_assembly*, *launch_ground_attack*, *launch_broadcasting*, *capture_bodyguard*, *use_special_force*. Just above decision nodes, we have chance nodes representing the outcomes of performing such actions (they indicate the workability of such systems), and below we have utility nodes (diamond-shaped nodes) describing the cost of each action. Furthermore, we have six chance nodes (in the center of the figure) indicating general workability of *IADS* (Integrated Air Defense System), *Air_force*, *Artillery*, *Ground_force*, *Morale* and *Commander_in_custody* with respect to enemy forces. The overall profit of winning is given by the node U_H , child of *Hypothesis*.

As this is an hypothetical example, we define utility functions and probability distributions as follows:

- Probability of *Hypothesis* is one given that all subgoals are achieved. If one of subgoals is not achieved, then the probability of *Hypothesis* is 60%; if two of them are not achieved, then the probability of success is 30%; if none of subgoals is achieved, then we certainly fail in the campaign.
- For the subgoals *Air_superiority*, *Territory_occupation*, and *Commander_surrender*, we define that the subgoal is accomplished with probability one when both children were achieved, 50% when only one child is achieved, and zero when none is achieved.
- For the probabilities of *IADS*, *Air_force*, *Artillery*, *Ground_force*, *Morale* and *Commander_in_custody*, we define a decrease of 50% for each unaccomplished child (with a minimum of zero, of course). Any node has probability zero if two or more of its children are not achieved.
- The outcomes of actions (chance nodes above decision nodes) have 90% of success. For exam-

Nodes		Approx.# of Strategies	CR			SPU	
Total	Decision		Time(sec)	Evals (B&B)	Max.Error(%)	Time(sec)	Max.Error(%)
10	3	2^{17}	0.66	5	0.000	0.10	0.740
20	6	2^{34}	1.73	125	0.000	0.39	2.788
50	10	2^{51}	30.42	4048	0.000	1.62	2.837
60	15	2^{52}	29.77	2937	0.000	2.99	1.964
70	20	2^{54}	125.06	7132	0.000	5.52	3.448
120	25	2^{102}	254.80	15626	0.544	11.58	2.193
120	30	2^{116}	403.13	5617	4.639	13.79	7.281
120	35	2^{120}	578.99	9307	5.983	16.87	11.584

Table 1: Average results on 30 random influence diagrams of different sizes for the CR and SPU methods.

ple, *destroy_Radars* will have *EW/GCI_radars* destroyed with 90% of odds (*EW/GCI* means *Early Warning/Ground Control Interception*).

- The reward of achieving the main goal is 1000, while not achieving it costs 500.
- Costs of actions are as follows: *ground_attack* is 150, *use_special_force* is 100, *capture_bodyguard* is 80, *air_strike* is 50, and other actions cost 20 each.

For this problem, the best strategy found by SPU has expected utility of -55.2825 , and suggests to take all action except *destroy_RD*, *destroy_storage*, *destroy_assembly* and *launch_ground_attack*. The global optimum strategy is found in less than 5 seconds with our method and has expected utility equal to 156.4051 (all actions are taken). This is much faster than the solution reported by [22] (around 45 seconds).

6 CONCLUSION

We discuss in this paper a new idea for strategy selection in Influence Diagrams. We work with the Limited Memory Influence Diagram, as it generalizes many of the influence diagram proposals. The main contribution is the reformulation of the problem as a credal network inference, which makes possible to find the global maximum strategy for small- and medium-sized influence diagrams. Experiments indicate that many instances can be treated exactly. As far as we know, no deep investigation of exact procedures for this class of diagrams has been conducted.

Because of the characteristics of our procedure, an anytime approximate solution with a maximum guaranteed error is available during computations. It is clear that large diagrams must be treated approximately. Nevertheless, in the conducted experiments, our method produced results that surpass existing algorithms. Although spending more time, many situations require a solution to be as good as possible,

while time is a secondary issue. The ability of our approach to provide an upper bound for the result is also valuable, which is not available with the SPU method.

We also discuss the theoretical complexity of the problem, which is derived from the known properties of MAP problems in Bayesian networks and belief updating inferences in credal networks. The complexity results show that the proposed idea is not making use of a harder problem to solve a simpler one, as the complexity of strategy selection is the same as the complexity of inferences in credal networks.

Because strategy selection in influence diagrams and inferences in credal networks are related, improvements on algorithms of credal networks can be directly applied to influence diagram problems. The application of other approximate techniques based on credal networks seems a natural path for investigation. We also intend to explore other optimization criteria for influence diagrams with imprecise probabilities, besides expected utility. Proposals in the theory of imprecise probabilities might be applied to this setting.

Acknowledgements

The work described in this paper is supported in part by the U.S. Army Research Office grant W911NF0610331.

References

- [1] A. Antonucci and M. Zaffalon. Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of Bayesian networks. *Int. J. Approx. Reason.*, in press, doi:10.1016/j.ijar.2008.02.005, 2008.
- [2] G. F. Cooper. A method for using belief updating as influence diagrams. In *Conf. on Uncertainty in Artif. Intelligence*, p. 55–63, Minneapolis, 1988.

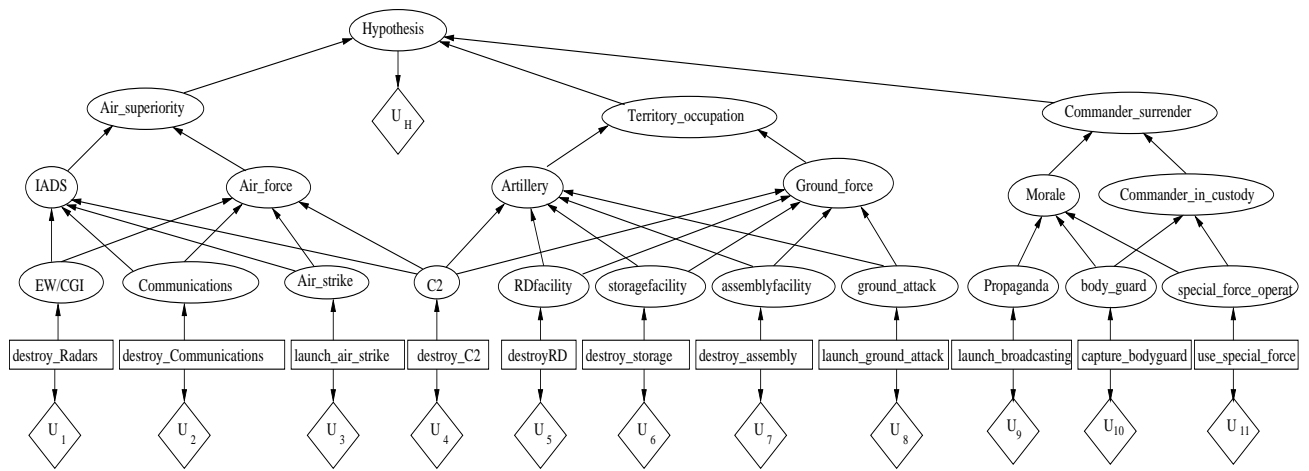


Figure 2: Influence Diagram for an hypothetical EBO-based planning problem.

- [3] I. Couso, S. Moral, and P. Walley. A survey of concepts of independence for imprecise probabilities. *Risk, Decision and Policy*, 5:165–181, 2000.
- [4] F. G. Cozman. Credal networks. *Artif. Intelligence*, 120:199–233, 2000.
- [5] F. G. Cozman. Separation properties of sets of probabilities. In *Conf. on Uncertainty in Artif. Intelligence*, p. 107–115, San Francisco, 2000.
- [6] P. Davis. Effects-based operations: a grand challenge for the analytical community. Technical report, Rand corp., 2003. MR1477.
- [7] C. P. de Campos and F. G. Cozman. Inference in credal networks using multilinear programming. In *Second Starting AI Researcher Symposium*, p. 50–61, Valencia, 2004. IOS Press.
- [8] C. P. de Campos and F. G. Cozman. The inferential complexity of Bayesian and credal networks. In *Int. Joint Conf. on Artif. Intelligence*, p. 1313–1318, 2005.
- [9] C. P. de Campos and F. G. Cozman. Inference in credal networks through integer programming. In *Int. Symp. on Imprecise Probability: Theories and Applications*, p. 145–154, 2007.
- [10] L. de Campos and S. Moral. Independence concepts for convex sets of probabilities. In *Conf. on Uncertainty in Artif. Intelligence*, p. 108–115, San Francisco, 1995.
- [11] D. A. Deptula. Effects-based operations: change in the nature of warfare. *Defense and Airpower Series*, p. 3–6, 2001.
- [12] E. Fagioli and M. Zaffalon. 2U: An exact interval propagation algorithm for polytrees with binary variables. *Artif. Intelligence*, 106(1):77–107, 1998.
- [13] R. A. Howard and J. E. Matheson. *Influence diagrams*, volume II, p. 719–762. Strategic Decisions Group, Menlo Park, 1984.
- [14] F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *Conf. on Uncertainty in Artif. Intelligence*, p. 367–373, San Francisco, 1994.
- [15] S. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1238–1251, 2001.
- [16] Ilog Optimization. Cplex documentation. <http://www.ilog.com>, 1990.
- [17] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artif. Intelligence Research*, 21:101–133, 2004.
- [18] R. Qi and D. Poole. A new method for influence diagram evaluation. *Computational Intelligence*, 11:1:1–34, 1995.
- [19] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.
- [20] N. L. Zhang. Probabilistic inferences in influence diagrams. In *Conf. on Uncertainty in Artif. Intelligence*, p. 514–522, Madison, 1998.
- [21] N. L. Zhang and D. Poole. Stepwise-decomposable influence diagram. In *Int. Conf. on Principles of Knowledge Representation and Reasoning*, p. 141–152, Cambridge, 1992.
- [22] W. Zhang and Q. Ji. A factorization approach to evaluating simultaneous influence diagrams. *IEEE Transactions on Systems, Man and Cybernetics A*, 36(4):746–757, 2006.

A Factorization Approach to Evaluating Simultaneous Influence Diagrams

Weihong Zhang and Qiang Ji, *Senior Member, IEEE*

Abstract—Evaluating an influence diagram (ID) is a challenging problem because its complexity increases exponentially in the number of decision nodes in the diagram. In this paper, we examine the problem for a special class of IDs where multiple decisions must be made simultaneously. We describe a brief theory that factorizes out the computations common to all policies in evaluating them. Our evaluation approach conducts these computations once and uses them across all policies. We identify the ID structures for which the approach can achieve savings. We show that the approach can be used to efficiently recompute the optimal policy of an ID when its structure or parameters change. Finally, we demonstrate the superior performance of the approach by simulation studies and a military planning example.

Index Terms—Algorithm, decision making under uncertainty, graphical model, influence diagram (ID), military analysis.

I. INTRODUCTION

AN INFLUENCE diagram (ID) is a plausible graphical model for decision making under uncertainty [1]. An ID comprises of decision nodes, random nodes, value nodes, and the probabilistic relations among these nodes. An ID is a more compact representation of a decision tree, which is a simple tool for decision analysis [2].

Given an ID, a policy prescribes an action choice for each decision node. Evaluating a policy is to compute the expected value of the ID under the policy. Evaluating an ID is to find the optimal policy that maximizes the expected value of the ID. A generic approach to evaluating an ID has to enumerate all policies, compare the expected utilities under them, and choose the optimal one. However, the number of policies grows exponentially with the number of decision nodes. This renders the approaches for general ID evaluation very inefficient and infeasible for large IDs. Consequently, it is advisable to study efficient algorithms for special IDs.

Most of the previous approaches assume that there exists a linear ordering among the decision nodes. This ordering implies that the choice of a decision node is known to the decision maker when he/she chooses the actions for the successive decision nodes (e.g., see [13]). For a decision node, this linear

ordering usually can be exploited to decompose the ID into one fraction prior to the node and the other fraction posterior to the node. The choice for the decision node can be made using the fraction posterior to the node. The procedure repeats for each decision node.

In this paper, we examine the ID evaluation problem for a special class of IDs in which decision nodes have no parents. Essentially, an ID with this property assumes no precedence relationship among decision nodes. In other words, one has to determine the choices for all decision nodes simultaneously. For this reason, such an ID is said to be simultaneous. The simultaneity assumption prevails in real-world problem domains. For instance, a military planner must select among a number of available actions to achieve his/her overall goal success; a business owner must consider multiple elements in order to maximize his/her monetary profit.

In evaluating a simultaneous ID, we exploit the assumption and divide the ID into two fractions, calling them the upstream and downstream. Roughly, the upstream consists of decision nodes and their children nodes through which the decisions propagate their impacts on the ID. Informally, these child nodes are called interface nodes. The downstream consists of the interface nodes and their succeeding nodes. We present a representation theorem, showing that the expected value of a value node under a policy can be represented as the sum of some intermediate quantities weighted by the probabilities determined by the policy. These intermediate quantities involve only the downstream. The factorization approach we proposed computes them once but uses them across all policies. The computational gain brought by the approach depends on the size of the downstream. Usually, larger downstream size implies more savings.

We organize the paper as follows. In the next section, we discuss related work to this research. We then introduce IDs and the evaluation problem. In Section IV, we describe the representation theorem and develop the factorization approach. In Section V, we discuss two extensions of the approach: how it can be adapted to network structure/parameter changes and how it can be used in planning over time. We report empirical results on simulation studies and a military planning example in Section VI. Finally, we conclude the paper in Section VII.

II. RELATED WORK

Since IDs were introduced by Howard and Matheson [1], a variety of approaches have been proposed to find the optimal policy of a given ID. To mitigate the exponential growth

Manuscript received February 19, 2004; revised July 20, 2004 and December 22, 2004. The work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Grant F49620-03-1-0160. Part of the work for this project was also supported by the Air Force Research Laboratory (AFRL)/Rome summer visiting faculty program. This paper was recommended by Associate Editor Yang.

The authors are with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180-3590 USA.

Digital Object Identifier 10.1109/TSMCA.2005.855753

problem of the policy number in the number of decision nodes, researchers have studied several special ID classes and proposed efficient approaches exploiting their specific problem characteristics. We give a brief survey of these IDs and their solutions.

A. Regular and No-Forgetting IDs

To some extent, most IDs that have been studied assume a precedence ordering of the decision nodes. A regular ID assumes that there is a directed path containing all decision nodes; a no-forgetting ID assumes that each decision node and its parents are also parents of the successive decision nodes; and a stepwise decomposable ID assumes that the parents of each decision node divide the ID into two separate fractions. These assumptions are different from ours, which requires the actions to be chosen simultaneously. There exist direct and indirect approaches evaluating a regular no-forgetting ID. A direct approach works on the ID and evaluates it directly. Shachter [3] proposed an algorithm that evaluates an ID by applying a series of value-preserving reductions. A value-preserving reduction is an operation that can transform an ID into another one with the same expected value. Specifically, Shachter identified the following four reductions: arc reversal, barren-node removal, random-node removal, and decision-node removal. An indirect approach first transforms an ID into an intermediate structure whose optimal policy (or value) remains the same as in the original ID. It then evaluates the intermediate structure and obtains the optimal policy. For instance, Howard and Matheson discussed a way to transform an ID into a decision-tree network and to compute an optimal policy from the decision tree. In transforming an ID into a decision-tree network, a basic operation is arc reversal [1], [3]. Since a no-forgetting ID must be stepwise decomposable, stepwise decomposability is more general than no-forgetting.

In most ID evaluation approaches, the ordering of decision nodes is an important information source in decision making and therefore, is exploited to evaluate the optimal decision for decision nodes [4]–[6]. A stepwise decomposable ID can be evaluated by a divide-and-conquer approach. The approach deals with one decision node at a time [7]. For each decision node, its parental set separates an ID into two parts—a body and a tail. The tail is a simple ID with only one decision node. The body's value node is a new value node whose value function is obtained by evaluating the tail. In evaluating a stepwise decomposable ID, the approach begins with a leaf decision node and repeats the decomposition/evaluation procedure for the preceding decision nodes. In evaluating the tail with only one single decision node, the problem is reduced to that of computing posterior probabilities in a Bayesian network. Hence, the approach uses probabilistic inference techniques to evaluate an ID. Cooper [8] initiated the research in this direction. He gave a recursive formula for computing the maximal expected utilities and optimal policies of IDs. Shachter and Peot [9] showed that the problem of ID evaluation can be reduced to a series of probabilistic inferences. Zhang [13] described an algorithm that induces much easier probabilistic inferences than those in [8] and [9].

B. Partial IDs

There also exists research work that relaxed the regularity or no-forgetting assumption. The specific ID types include partial IDs, unconstrained IDs and limited memory ID (LIMID), which is a compact representation of IDs. A partial ID is an ID that allows a non-total ordering of decision nodes [10]. Because the solution to a partial ID depends on the temporal ordering of the decisions, it is of interest to find the conditions identifying a class of partial IDs whose solution is independent of the legal evaluation ordering. Based on the concept of d-connectivity, Nielsen and Jensen presented an algorithm determining whether or not a partial ID represents well-defined scenarios, and they also addressed the problem of whether all admissible orderings yield the same optimal strategy.

An unconstrained ID is an ID where the order of decision nodes and the observable random nodes is not determined [11]. For an unconstrained ID, it is of interest to determine the order of decision nodes and information on which set of nodes is necessary for decision making in a decision node. For this purpose, a set of rules have been developed in order to determine the choice of the next decision node, given the current information. Such a decision choice may be dependent on the specific information from the past.

Another recently proposed ID is called LIMID, which violates the no-forgetting assumption [12]. In contrast to the regular and no-forgetting assumption, the assumption behind a LIMID is that only requisite information for the computation of optimal policies is depicted in the graphical representation. Two properties pertaining to LIMIDs are: 1) any ID can be converted to a LIMID; and 2) the converted LIMID is more compact than the original ID in the sense that only requisite information is depicted in the LIMID for computing an optimal policy. By these properties, one may convert an ID to its LIMID version and solve the LIMID instead of the original ID. This optimal policy is also optimal in the original ID. The algorithm solving a LIMID exploits the fact that the entire decision problem can be partitioned into a set of smaller decision problems, each of which has one decision node only. This is analogous to the divide-and-conquer approach [13].

C. Simultaneous IDs

From its root definition, an ID does not impose a precedence ordering of the decision nodes. As an example, there are military applications that need to choose multiple actions simultaneously. A simultaneous ID is suitable for this situation. We exploit this assumption and divide a simultaneous ID into the upstream and the downstream fractions. The decomposition takes the random and value nodes as interface nodes between the upstream and the downstream. The computations involving the downstream fraction can be precomputed and reused across all policies in evaluating them. This computation-sharing schema can greatly accelerate the procedure of finding the optimal policy for a given ID, as indicated in our theoretical and empirical analysis.

Technically, the factorization approach has some conceptual similarities to the probabilistic inference-based algorithm [13].

Both algorithms divide the ID into two fractions. However, there are apparent differences. In [13], the separation of an ID relies on a single decision node. With respect to a decision node, roughly, the body contains the predecessors of the decision nodes, while the tail contains the successors. The choice of the decision node is evaluated by the tail part. This is quite different from our factorization approach, where the separation relies on the set of interface nodes. The set of the interface nodes separates an ID into two fractions: roughly, the upstream contains the predecessors of all interface nodes, while the downstream contains the successors. This difference in solution techniques stems from the difference in assumption—the probabilistic inference algorithm works with a regular ID that specifies a linear order among decision nodes, whereas the factorization algorithm works with a simultaneous ID that assumes no ordering among decision nodes.

III. INFLUENCE DIAGRAM

Mathematically, an ID \mathcal{I} is a directed acyclic graph consisting of three types of nodes and the links among these nodes [1].

- 1) Its node set is partitioned into a set of random nodes \mathcal{Y} , a set of decision nodes \mathcal{X} , and a set of value nodes \mathcal{U} . A value node cannot have children. The links characterize the conditional dependence among the nodes in the ID. Specifically, links to a random node indicate the probabilistic dependence of the node on its parents; links to a decision node indicate the information available to the planner at the time the planner must choose a decision for it; and links to a value node indicate the functional dependencies.

We will adopt the following notational conventions. We will use bold-typed letters such as \mathcal{Z} to denote a set of variables and capital letters such as Z to denote a variable in the set. Each random or decision node Z is associated with a set Ω_Z , denoting the set of its possible states. The set Ω_Z is called the domain of node Z . An element in Ω_Z is denoted by a low-case letter z . For any node Z , we use $\pi(Z)$ to denote its parent set. For any subset $\mathcal{Z}' \subset \mathcal{Y} \cup \mathcal{X}$, we use $\Omega_{\mathcal{Z}'}$ to denote the Cartesian product $\prod_{Z \in \mathcal{Z}'} \Omega_Z$. For convenience, we shall interchangeably use a node and a variable. Without loss of generality, we assume that all the nodes are binary throughout this paper.

- 2) For each decision or random node Z , given an assignment of $\pi(Z)$, the distribution $P(Z|\pi(Z))$ specifies the probability of Z being in each state of the node Z . Such a distribution is called a conditional probability table (CPT) in the case that the domain of the variable Z is a finite set.
- 3) For each value node U , g_U is a value function $g_U : \Omega_{\pi(U)} \rightarrow R$, where R denotes the set of the real numbers.

To avoid unnecessary notations, we define the (optimal) policy concept only for a simultaneous ID.¹ A policy, denoted by δ , specifies one action choice for each decision node in \mathcal{X} . Hence, a policy δ can be denoted by $(\delta_1, \dots, \delta_n)$, where δ_i belongs to the domain of X_i for each i .

¹For general IDs, the definition of an (optimal) policy can be found in, e.g., [6].

Given a policy δ , a probability P_δ can be defined over the random nodes and decision nodes as follows:

$$P_\delta(\mathcal{Y}, \mathcal{X}) = \prod_{Y \in \mathcal{Y}} P(Y|\pi(Y)) \prod_{i=1}^n P_\delta(X_i) \quad (1)$$

where $P(Y|\pi(Y))$ is specified in the definition of \mathcal{I} , while $P_\delta(X_i)$ is equal to 1.0 if $X_i = \delta_i$, and 0.0 otherwise.

The expectation of the value node U under policy δ , denoted by $E_\delta[U]$, is defined as

$$E_\delta[U] = \sum_{\pi(U)} P_\delta(\pi(U)) g_U(\pi(U)). \quad (2)$$

The expected value E_δ of \mathcal{I} under the policy δ is the sum $E_\delta[U]$ over all value nodes U in \mathcal{U} , i.e.,

$$E_\delta = \sum_{U \in \mathcal{U}} E_\delta[U]. \quad (3)$$

For simplicity, E_δ is also called the expected value of policy δ . Evaluating a policy δ means to compute its expected value. The maximum of E_δ over all policies is the optimal (expected) value of \mathcal{I} . An optimal policy is the policy that achieves the optimal expected value. To evaluate an ID is to find an optimal policy and to compute its optimal expected value.

IV. THE FACTORIZATION APPROACH

In this section, we describe the representation theorem and the factorization approach.

A. The Idea

From its definition, an ID is a network structure consisting of decision nodes, random nodes, and value nodes. Among them, in determining the expected value of the ID, a decision node plays a different role from a random or a value node. The choices of a decision node can affect the expected value of the ID through changing the CPTs of its child random nodes, or through changing the value functions of its child value nodes (note that a decision node cannot have another decision node as child in a simultaneous ID). In this sense, a node, if it is a child of a decision node, serves as an interface through which the choices of decision nodes may affect the value of the ID. Such a node is called an interface node. All interface nodes constitute an interface set. Collectively, an interface set serves as an interface of an ID through which policies can affect the expected value of the ID. Consequently, an ID can be divided into two fractions: the upstream fraction, which includes the interface nodes and the nodes “preceding” them, and the downstream fraction, which includes the interface nodes and the nodes “succeeding” the interface nodes.

Example: We use the ID in Fig. 1 to informally illustrate these concepts. The ID has two decision nodes $\{X_1, X_2\}$, five random nodes $\{A, B, C, D, H\}$, and one value node U . The interface set \mathcal{Y}_{in} is $\{A, C\}$ since they have parental decision nodes. The upstream is $\{X_1, X_2, A, B, C\}$, which consists of two interface nodes A and C , node X_1 preceding node A , and nodes B and X_2 preceding node C . The downstream is

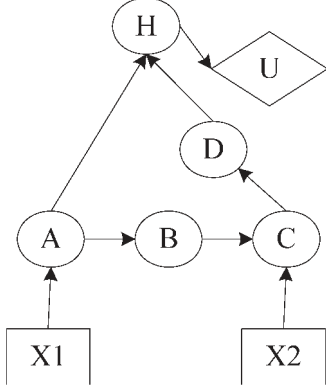


Fig. 1. ID to illustrate the representation theorem.

$\{A, C, H, D, U\}$, which consists of interface nodes A and C , the nodes H, D, U succeeding to the interface nodes. ■

Interestingly, corresponding to the structural separation that an ID can be divided into two fractions, the expected value of a value node under a policy breaks into two fractions, each of which involving only the upstream or the downstream of the ID.

B. The Theorem

We formalize the above idea in this section. For the sake of simplicity, throughout the paper, unless explicitly stated, we assume that: 1) the ID has only one value node; and 2) the value node has no decision node as its parent. We also note that our results in this paper generalize to the IDs with multiple value nodes and with value nodes having parental decision nodes. We relax these assumptions at the end of this section.

We begin by defining several concepts. A random node Y is an interface node if its parent set has at least one decision variable, i.e., $\pi(Y) \cap \mathcal{X} \neq \emptyset$. The interface set of an ID is the set of all interface nodes. Due to the above assumptions, the interface set contains only random nodes; for this reason, we denote the set by \mathcal{Y}_{in} . The upstream of the ID includes the interface set and all ancestors of the nodes in the interface. By this definition, in addition to the interface random nodes and decision nodes, the upstream may contain the random-node ancestors of the interface nodes. These ancestral nodes must be included because they, together with decision nodes, determine the CPTs of the interface nodes. These ancestral random nodes form a set denoted by \mathcal{Y}_0 .

Given an ID, we can efficiently identify its upstream using a queuing mechanism. We initialize a queue to be the interface set \mathcal{Y}_{in} (it can be readily built by checking whether there is a parental decision node for every node in the ID) and the upstream \mathcal{I}_{up} to be empty. At each step, a node is removed from the queue and added to \mathcal{I}_{up} if it is not in \mathcal{I}_{up} . The parents of the node, if not present in \mathcal{I}_{up} thus far, are added to the queue. The procedure terminates when the queue is empty. When it terminates, the set \mathcal{I}_{up} becomes the upstream set. The procedure must terminate after a finite number of steps because an ID is a directed acyclic graph.

The upstream can be partitioned into three sets: the set \mathcal{X} of decision nodes, interface set \mathcal{Y}_{in} , and the set \mathcal{Y}_0 of random-node ancestors of interface nodes. Given a policy δ , we define

a function f_δ from $\Omega_{\mathcal{Y}_{\text{in}}}$ to the real line R . For notations, we let m be the number of nodes in the set \mathcal{Y}_{in} , $Y_{\text{in}}^{1:m}$ be a short notation of $\{Y_{\text{in}}^1, \dots, Y_{\text{in}}^m\}$, and $y_{\text{in}}^{1:m}$ be an assignment to all interface variables, i.e., an element of $\Omega_{\mathcal{Y}_{\text{in}}^{1:m}}$

$$f_\delta(y_{\text{in}}^{1:m}) = \sum_{Y \in \mathcal{Y}_0} \prod_{Y \in \mathcal{Y}_0 \cup \mathcal{Y}_{\text{in}}} P_\delta(Y|\pi(Y)) \prod_{i=1}^n P_\delta(X_i) \quad (4)$$

where $\prod_{Y \in \mathcal{Y}_0 \cup \mathcal{Y}_{\text{in}}} P_\delta(Y|\pi(Y)) \prod_{i=1}^n P_\delta(X_i)$ is the joint probability distribution of the variables in \mathcal{X} , \mathcal{Y}_0 , and \mathcal{Y}_{in} , given policy δ . Hence, $f_\delta(Y_{\text{in}}^{1:m})$ is the conditional probability that the interface $Y_{\text{in}}^{1:m} = y_{\text{in}}^{1:m}$ occurs upon the policy δ . For convenience, we call them interface probabilities.

In contrast to the upstream, the downstream of an ID is the set consisting of all the interface nodes and their descendants. The downstream contains the value node, the interface nodes, and the random nodes that do not belong to the upstream. We use \mathcal{Y}_1 to denote the set of noninterface random nodes in the downstream. Note that the random nodes in the interface set belong to both the upstream and the downstream. For an assignment $y_{\text{in}}^{1:m}$ of the set $Y_{\text{in}}^{1:m}$ and the value node U , we can define a function as follows;

$$f_{\text{in},U}(y_{\text{in}}^{1:m}) = \sum_{Y \in \mathcal{Y}_1} \prod_{Y \in \mathcal{Y}_1} P_\delta(Y|\pi(Y)) g_U(\pi(U)). \quad (5)$$

To see that $f_{\text{in},U}$ is a function of $Y_{\text{in}}^{1:m}$, we note that the interface variables may appear in $\pi(Y)$ for $Y \in \mathcal{Y}_1$. Given an assignment $y_{\text{in}}^{1:m}$ of $Y_{\text{in}}^{1:m}$, $f_{\text{in},U}(y_{\text{in}}^{1:m})$ is the expected utility conditioned on the assigned interface $y_{\text{in}}^{1:m}$. These quantities are called interface utilities for convenience. Since $\pi(Y)$ for Y in \mathcal{Y}_1 must belong to the downstream, $P_\delta(Y|\pi(Y))$ is independent of policy δ . Consequently, these utilities are independent of policy δ .

Theorem 1: Given a policy δ and a value node U , the expected value of the node U under policy δ

$$E_\delta[U] = \sum_{y_{\text{in}}^{1:m} \in \Omega_{\mathcal{Y}_{\text{in}}^{1:m}}} f_\delta(y_{\text{in}}^{1:m}) \cdot f_{\text{in},U}(y_{\text{in}}^{1:m}). \quad (6)$$

Proof: We show that $E_\delta[U]$ can be rewritten as the sum of the interface utility $f_{\text{in},U}$ weighted by the probability f_δ over all interfaces

$$E_\delta[U] = \sum_{\pi(U)} P_\delta(\pi(U)) g_U(\pi(U)) \quad (a)$$

$$= \sum_{\pi(U)} \left[\sum_{\frac{\mathcal{Y}}{\pi(U)}} \prod_{Y \in \mathcal{Y}} P(Y|\pi(Y)) \prod_{i=1}^n P_\delta(X_i) \right] \times g_U(\pi(U)) \quad (b)$$

$$= \sum_{Y \in \mathcal{Y}_{\text{in}}} \sum_{Y \in \mathcal{Y}_0} \sum_{Y \in \mathcal{Y}_1} \prod_{Y \in \mathcal{Y}_0 \cup \mathcal{Y}_{\text{in}}} P(Y|\pi(Y)) \times \prod_{Y \in \mathcal{Y}_1} P(Y|\pi(Y)) \prod_{i=1}^n P_\delta(X_i) g_U(\pi(U)) \quad (c)$$

$$\begin{aligned}
 &= \sum_{Y \in \mathcal{Y}_{\text{in}}} \left[\sum_{Y \in \mathcal{Y}_0} \prod_{Y \in \mathcal{Y}_0 \cup \mathcal{Y}_{\text{in}}} P(Y|\pi(Y)) \right] \\
 &\quad \times \left[\sum_{Y \in \mathcal{Y}_1} \prod_{Y \in \mathcal{Y}_1} P(Y|\pi(Y)) g_U(\pi(U)) \right] \quad (\text{d}) \\
 &= \sum_{y_{\text{in}}^{1:m} \in \Omega_{Y_{\text{in}}^{1:m}}} f_{\delta}(y_{\text{in}}^{1:m}) \cdot f_{\text{in},U}(y_{\text{in}}^{1:m}). \quad (\text{e})
 \end{aligned}$$

Step (a) is true by (2). At step (b), $\mathcal{Y}/\pi(U)$ is the difference set of \mathcal{Y} and $\pi(U)$. This step is true by inserting (1) into (2). Step (c) follows from the fact that $\{\pi(U), \mathcal{Y}/\pi(U)\}$ and $\{\mathcal{Y}_0, \mathcal{Y}_{\text{in}}, \mathcal{Y}_1\}$ are two partitions of the set \mathcal{Y} . At step (d), we break the distribution $\prod_{Y \in \mathcal{Y}} P(Y|\pi(Y)) \times \prod_{i=1}^n P_{\delta}(X_i)$ into two fractions $\prod_{Y \in \mathcal{Y}_0 \cup \mathcal{Y}_{\text{in}}} P(Y|\pi(Y))$ and $\prod_{Y \in \mathcal{Y}_1} P(Y|\pi(Y)) g_U(\pi(U))$. At step (e), we replace the two fractions with the definitions of the interface utilities and interface probabilities. ■

By the theorem, given a policy δ and a value node U , the expected value of the node under the policy can be represented as the sum of the multiplications of the interface utilities and corresponding interface probabilities.

Example (Continued): For the ID in Fig. 1, we show how to represent $E_{\delta}[U]$ for the value node U and a given policy δ . Let the policy δ be (δ_1, δ_2) , where δ_i is the decision choice of X_i for $i = 1, 2$. By definition

$$\begin{aligned}
 E_{\delta}[U] &= \sum_H \sum_{A, B, C, D} P(A|\delta_1 \delta_2) P(B|A) P(C|B \delta_2) \\
 &\quad \times P(D|C) P(H|AD) \prod_{i=1}^2 P_{\delta}(X_i) g_U(H).
 \end{aligned}$$

The two functions are defined as follows.

$$\begin{aligned}
 f_{\delta}(A, C) &= \sum_B P(A|\delta_1 \delta_2) P(B|A) P(C|B \delta_2) \prod_{i=1}^2 P_{\delta}(X_i) \\
 f_{\text{in},U}(A, C) &= \sum_{H, D} P(H|AD) P(D|C) g_U(H).
 \end{aligned}$$

It can be verified that $E_{\delta}[U] = \sum_{A, C} f_{\delta}(A, C) \cdot f_{\text{in},U}(A, C)$. ■

We examine the assumptions we made at the beginning of this section. First, we have assumed that there is only one value node. In case of multiple value nodes, we may apply the representation theorem to each node. The expected value of a policy is the additive sum of the expected values of all value nodes under the policy.

Second, we have assumed that the value node has no decision nodes as its parents. In the other case that the value node has a decision node as its parents, the functions $f_{\delta,U}$ and f_{in} can be defined as follows, such that the theorem holds

$$\begin{aligned}
 f_{\delta,U}(Y_{\text{in}}^{1:m}) &= \sum_{Y \in \mathcal{Y}_0} \prod_{Y \in \mathcal{Y}_0 \cup \mathcal{Y}_{\text{in}}} P_{\delta}(Y|\pi(Y)) \\
 &\quad \times \prod_{i=1}^n P_{\delta}(X_i) g_U(\pi(U)) \\
 f_{\text{in}}(Y_{\text{in}}^{1:m}) &= \sum_{Y \in \mathcal{Y}_1} \prod_{Y \in \mathcal{Y}_1} P_{\delta}(Y|\pi(Y)).
 \end{aligned}$$

TABLE I
FACTORIZATION APPROACH TO ID EVALUATION

- | | |
|-----|---|
| 1. | Pre-compute the quantities $f_{\text{in},U}(y_{\text{in}}^{1:m})$ for all $y_{\text{in}}^{1:m}$ in $\Omega_{Y_{\text{in}}^{1:m}}$ |
| 2. | For each policy δ |
| 2.1 | compute $f_{\delta}(y_{\text{in}}^{1:m})$ for each assignment of $Y_{\text{in}}^{1:m}$ |
| 2.2 | compute $E_{\delta}[U]$ by Equation (6) |
| 3. | Return the policy that maximizes $E_{\delta}[U]$ |

Therefore, we can lift the assumption that the value node has no decision node as parents. In this case, U is also called an interface node, but it belongs to the upstream only. The reason is that, by definition, a value node cannot have children and therefore cannot produce impact on the downstream.² Note that f_{δ} changes to $f_{\delta,U}$, since the value node is considered in computing the quantities relevant to the upstream. Interestingly, it can be proven that $f_{\text{in}} (= 1.0)$ is a constant. To see why, let us assume that the size of \mathcal{Y}_1 be k . We enumerate the set \mathcal{Y}_1 as $\{Y_1^1, \dots, Y_1^k\}$ such that a node's parents appear after the node in the set. In computing $f_{\text{in}}(Y_{\text{in}}^{1:m})$, we can sequentially sum out the variables in \mathcal{Y}_1 in the enumerated order. Ultimately, we have $f_{\text{in}} = 1.0$.

C. The Algorithm

By the representation theorem, the expected value of a policy is represented as the sum of interface utilities weighted by the corresponding interface probabilities. The interface utilities are independent of the individual policies, whereas the interface probabilities are dependent on the policies. Therefore, the interface utilities can be factored out, i.e., they can be calculated once and reused across all the policies.

This is the idea behind our factorization approach, which is described in Table I. The factored-out computations are calculated once at line 1. They are used for all policies at line 2.2. Note that the procedure generalizes to IDs with multiple value nodes.

D. Complexity Analysis

It is of interest to compare the approach and the generic brute-forced approach that evaluates a policy directly by combining (1) and (2). Let n be the number of decision nodes. Thus, the size of the policy space is 2^n . Let the complexity of evaluating one policy be C . The complexity C breaks into three pieces: computing f_{δ} , computing $f_{\text{in},U}$, and computing $E_{\delta}[U]$ by (6). We denote them, respectively, by C_1 , C_3 , and C_2 . To evaluate all policies, the generic approach has complexity $2^n C$, i.e., $2^n(C_1 + C_2 + C_3)$. In contrast, since the factorization approach computes $f_{\text{in},U}$ only once but uses them 2^n times, its complexity is $2^n(C_1 + C_2) + C_3$. A good measure to predict the computational gain is the size of the downstream, i.e., the number of nodes in it. In one extreme, if the downstream contains only the interface nodes and the value nodes (thus, C_3 is a constant), the two approaches have the same complexity. In the other extreme, if the downstream contains far more nodes

²Note that this is different from random nodes having parental decision nodes. Such a random node belongs to both the upstream and the downstream.

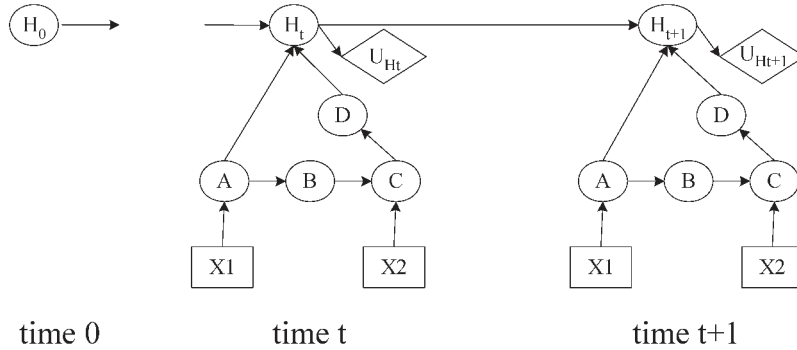


Fig. 2. Dynamic ID model.

than the upstream (i.e., $C_3 \gg C_1 + C_2$), the computational gain is significant.

E. Bounding the Optimal Expected Value

We show that the interface utilities computed in the factorization approach can be used to derive both an upper and a lower bound of the optimal value of the ID. These bounds have significant implications in practical planning.

We define $f_{in,U}^+$ and $f_{in,U}^-$ to be the largest and the smallest one among all interface utilities, i.e.,

$$f_{in,U}^+ = \max_{y_{in}^{1:m} \in \Omega_{Y_{in}^{1:m}}} f_{in,U}(y_{in}^{1:m})$$

$$f_{in,U}^- = \min_{y_{in}^{1:m} \in \Omega_{Y_{in}^{1:m}}} f_{in,U}(y_{in}^{1:m})$$

where the max and min are taken over the domain of the variables in $Y_{in}^{1:m}$. From the theorem, we see that $f_{in,U}^+$ ($f_{in,U}^-$) is the upper (lower) bound of the optimal value of the ID. These bounds have significant importance in practice. Suppose, for instance, that these bounds are available to a planner. In one extreme, if the planner expects a utility that is larger than the upper bound, he never bothers to evaluate all the policies and finds the optimal one because even the best policy provides less than he expects. In this case, he needs to redesign the network structure or parameters such that the performance of the ID can be improved. In the other extreme, if the planner expects a utility that is less than the lower bound, again he never bothers to evaluate all the policies and chooses the optimal one because any policy can provide more than he expects. In this case, he can pick any policy and execute it.

We note that from the computational point of view, computing these bounds is easier than evaluating the ID. There are two reasons. First, as discussed earlier, computing these bounds involves only the downstream of the ID, whereas evaluating an ID involves its entire structure. If the downstream contains much fewer variables than the upstream, the interface utilities (and also the bounds) can be obtained efficiently. Second, computing these bounds avoid enumerating all the policies and calculating their expected values.

Finally, the tightness of the bounds depends on the structure of an ID, the CPTs of random nodes, and the value functions of value nodes. It is difficult to characterize a general condition to determine the tightness of the bounds. In our experiments, we

empirically show that these bounds are reasonably tight for the tested problems.

V. EXTENSIONS TO THE FACTORIZATION APPROACH

In this section, we discuss two extensions to the factorization approach. These extensions deal with reconstructing the policies as the network structure/parameters undergo changes. There are two perspectives. First, at one decision step, the network might change such as more actions being available for a planner's choice, more value nodes needed consideration, and so on. Second, the network might dynamically alter its structure or parameters as time goes by. For example, if a subgoal is successfully accomplished at one step, it can be removed from the network in the subsequent steps.

A. Network Structure/Parameter Changes

The principle for the factorization approach to accommodate structure or parameter changes is as follows. First, if the changes involve only the upstream of an ID, the interface utilities do not need to be recomputed and can still be shared in evaluating the ID. Specifically, these changes include addition or removal of decision/random/value nodes and also the alternation of CPTs and value functions in the upstream. Second, if the changes involve only the downstream of the ID, the approach needs to reconstruct the interface utilities. Fortunately, the interface probabilities are preserved and the calculations for them can be saved. Third, if the changes involve not only the upstream but also the downstream, the approach needs to recompute both the interface utilities and the interface probabilities.

B. Planning Over Time

In realistic applications, network parameters may change over time. In this case, we can use a dynamic ID to model the conditional dependencies among nodes over time. In this section, we show how the factorization approach can be used to reconstruct the policies on a step-by-step basis for dynamic IDs.

To facilitate our discussions, we extend the example in Fig. 1 to a dynamic ID. We assume that the variable H evolves over time and let H_t denote H at step t . The dynamic ID is drawn in Fig. 2. In contrast, the ID in Fig. 1 is said to be static since the multiple decisions are made at one time step.

The dynamic ID has two prominent features. First, at a single step, the decision problem can be modeled as a static ID. In addition to the nodes and links in Fig. 1, the node H_{t+1} at step $t + 1$ has one more parent node H_t . Second, the intertemporal link between two consecutive nodes carries the historic information about the sequence of performed policies. For step $n + 1$, the information can be summarized by a probability distribution of H_t conditioned on the history [14].

For a dynamic ID, we are interested in optimal planning on the step-by-step basis. The problem is formulated as: Given an initial probability distribution $P(H_0)$, at step $t + 1$, how to efficiently find the policy $\delta = (\delta_1, \dots, \delta_n)$ where n is the number of decision nodes] that maximizes $E_\delta[U_{H_{t+1}}]$? To solve the problem, we show: 1) how to select the optimal policy at step $t + 1$, given the probability distribution $P(H_t)$; and 2) how to sequentially update the probability distribution $P(H_{t+1})$ from $P(H_t)$, given a policy δ at the previous step. After these two questions are settled, we may choose the optimal policy as follows. At step $t + 1$, we first choose the optimal policy for the step and then update the probability $P(H_{t+1})$ from $P(H_t)$. The procedure repeats at each step.

To answer the first question, we introduce the concept of an augmented interface node and an augmented interface set. We call the node H_t an augmented interface node of the ID at step $t + 1$ since the node H_t can produce impact on the network via altering its probability distribution. In this sense, it is an interface node.³ The augmented interface set consists of \mathcal{Y}_{in} as before and the node H_t . The downstream of the ID at step $t + 1$ remains the same as that of the static ID. Likewise, we may define the two functions f_δ and $f_{\text{in}, U_{H_{t+1}}}$. Therefore, we can use the factorization approach to solve the planning problem over time. The computations involving $f_{\text{in}, U_{H_{t+1}}}$ are factored out. Note that these interface utilities are shared for all policies at each decision step. For the ID in Fig. 2, we can define the following functions for the ID:

$$\begin{aligned} f_\delta(A, C, H_t) &= P(H_t) \sum_B P(A|\delta_1\delta_2)P(B|A)P(C|B\delta_2)\prod_{i=1}^2 P_\delta(X_i) \\ f_{\text{in}, U_{H_t}}(A, C, H_t) &= \sum_{H_{t+1}D} P(H_{t+1}|ADH_t)P(D|C)g_V(H_{t+1}). \end{aligned}$$

It can be verified that $E_\delta[U_{H_{t+1}}] = \sum_{A,C,H_t} f_\delta(A, C, H_t) \cdot f_{\text{in}, U_{H_t}}(A, C, H_t)$.

To answer the second question, we show how to efficiently compute $P(H_{t+1})$, given a distribution $P(H_t)$ and the policy δ performed at step t . We introduce a technique such that the procedure of computing $P(H_{t+1})$ can be conducted similar to that of computing $E_\delta[U_{H_{t+1}}]$. Suppose that H_{t+1} can take on two values $h(\text{true})$ and $\neg h(\text{false})$. We first show how to calculate

the probability of H_{t+1} being true. Let V be a value node that differs from $U_{H_{t+1}}$ only in its value function. Specifically, g_V is 1.0 if its parent H_{t+1} is true; it is 0.0 otherwise. For simplicity, let $H_{t+1} = h(\neg h)$ denote the event that the hypothesis H_{t+1} is true (false). We prove that $E_\delta[V] = P_\delta(H_{t+1} = h)$.

Proposition 1: $E_\delta[V] = P_\delta(H_{t+1} = h)$.

Proof:

$$\begin{aligned} E_\delta[V] &= \sum_{H_{t+1}} P_\delta(H_{t+1})g_V(H_{t+1}) \\ &= P_\delta(H_{t+1} = h)g_V(H_{t+1} = h) \\ &\quad + P_\delta(H_{t+1} = \neg h)g_V(H_{t+1} = \neg h) \\ &= P_\delta(H_{t+1} = h). \end{aligned}$$

In the last step, we use the definition of the value function g_V . ■

To calculate the probability of H_{t+1} being false, we may define g_V as follows: It is 1.0 if its parent H_{t+1} is false; it is 0.0 otherwise. If we define two functions f_δ and $f_{\text{in}, V}$, we see that the computational steps for $E_\delta[V]$ are the same as those for computing $E_\delta[U_{H_{t+1}}]$. Hence, computing $P(H_{t+1})$ does not add much overhead to ID evaluation.

It is interesting to compare the generic approach and the factorization approach in the context of the dynamic ID. Let the number of decision steps be T . Recall that the complexity of computing f_δ is C_1 , the complexity of computing $f_{\text{in}, U_{H_{t+1}}}$ is C_3 , and the complexity of computing $E_\delta[U_{H_{t+1}}]$ is C_2 . Since C_1 takes constant time, it can be ignored. For one decision step, the factorization approach has the complexity $2^n C_2 + C_3$ while the generic approach has the complexity $2^n(C_2 + C_3)$. For T steps, the complexity of the factorization approach is $2^n T C_2 + C_3$ [note that this does not include the overhead of computing $P(H_{t+1})$], while the complexity of the generic approach is $2^n T(C_2 + C_3)$. If $C_3 \gg C_2$, the factorization approach can be extremely efficient.

VI. EXPERIMENTS

In this section, we report our experiments on both simulation studies and a military planning example. In our experiments, we wrote Matlab-V6.5 codes and ran them on a laptop with a 2.0-GHz central processing unit (CPU) under Windows XP. We compare the factorization approach against the generic brute-forced approach. We chose the generic approach because we were not aware of specific algorithms for evaluating simultaneous IDs. For convenience, we refer to the two algorithms as `evalCS` (named after computation sharing) and `evalBF` (named after brute forced).

A. Simulation Studies

To thoroughly evaluate the performance of the factorization approach, we conducted simulated studies on the ID in the left chart of Fig. 3, which is similar to the military planning examples in [15]. It is referred to as the static ID in the rest of this section. The CPTs are randomly generated. The value functions for value nodes are manually specified.

Specifically, our experiments are designed to: 1) evaluate the performances of the factorization approach for static and

³Previously, we defined an interface node to be a node that has parental decision nodes since the choices of decision nodes can affect its CPTs and in turn, the expected value of the ID. In contrast, the node H_t is called an augmented interface node since it can change its probability distribution and thus, affect the expected value of the ID.

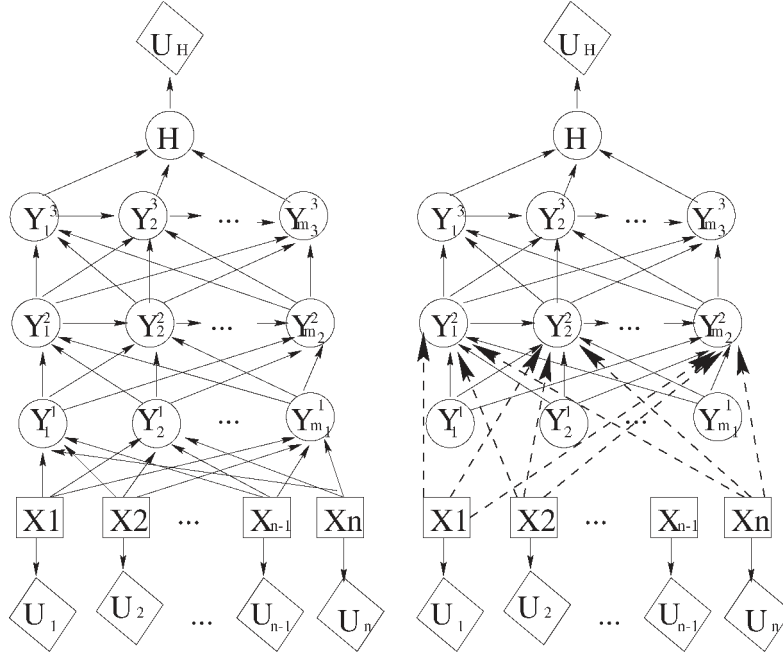


Fig. 3. Test example is shown in the left chart, while the right is its variant for comparative studies.

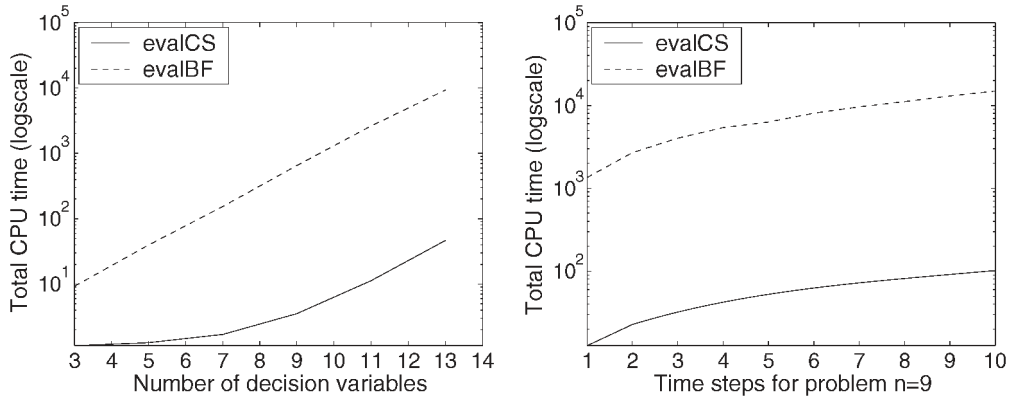


Fig. 4. Performance comparison of evalBF and evalCS.

dynamic IDs; 2) show the tightness of bounds derived from the interface utilities; 3) demonstrate how the computational gain achieved by the approach varies with different network structures; and 4) demonstrate the computational gain by adapting the approach to account for newly added decisions and value nodes.

1) *Performances of the Factorization Approach:* To see how the performances of the algorithms vary with the number of decision nodes, we fix the number of random nodes at each level at four and vary the number of decision nodes. Thus, the static ID with n decision nodes has additionally ten random nodes and $n + 1$ value nodes. We ran evalBF and evalCS for seven problems with $n = 3, 5, \dots, 13$. The timing data are presented in the left chart of Fig. 4. The chart gives the total CPU seconds that the algorithms took for each of the problems. Note that the vertical direction is drawn in log scale. The solid (dashed) curve is for evalCS (evalBF). It can be seen that evalCS is considerably more efficient than evalBF. For instance, from our data, for $n = 9$, to evaluate 512 policies,

evalCS took 3.51 s while evalBF, 646.74 s; for $n = 13$, to evaluate 8192 policies, evalCS used 46.32 s while evalBF, 9284.05 s.

To quantitatively characterize how much savings the factorization procedure can bring about, we use the timing results of evalCS to predict the performance of evalBF. Recall that the complexity of evaluating a policy breaks into three fractions C_1 , C_3 , and C_2 . We ignore C_1 since it is a constant. For each problem, we estimate C_3 by the actual seconds \hat{C}_3 of computing f_{in,U_H} , and C_2 by \hat{C}_2 as $(\text{the total CPU time} - \hat{C}_3) / (\text{the number of policies})$. The complexity of evalBF is predicted by $2^n(\hat{C}_2 + \hat{C}_3)$. We found that these estimations are almost the same as the actual timing results of evalBF. This suggests the effectiveness of our complexity analysis.

We also tested the algorithms over the dynamic ID in Fig. 5, which is an extension of the left chart of Fig. 3.

Initially, the probability of the node H being false is set to 1.0. Its probability is updated at each decision step. We ran both algorithms for up to ten decision steps. We showed the

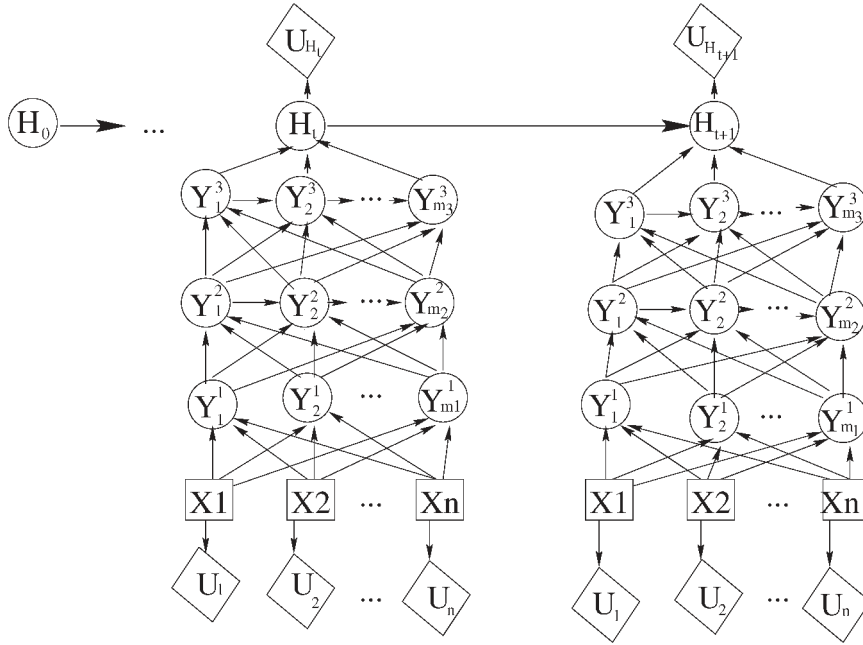


Fig. 5. Dynamic influence diagram.

total CPU time for the ID with nine decision nodes in the right chart of Fig. 4. The chart gives the total CPU seconds for both algorithms against the time steps. Note that again the vertical direction is drawn in log scale. It can be seen that the CPU time linearly increases with the elapsed time for *evalBF* while its increase is negligible for *evalCS*. This is not a surprising observation. In *evalBF*, all policies are evaluated at each step. The time cost for all steps remains the same. Hence, the increase is linear. From our data, *evalBF* uses about 1300 s to evaluate all 512 (2^9) policies at each step. However, in *evalCS*, the interface utilities are computed only once at the first step. So, we observe that at the first step, *evalCS* takes about 2.66 s to compute these utilities; thereafter, each step takes about only 10 s to evaluate all 512 policies. The increase is negligible when compared against that in *evalBF*.

2) *Tightness of Bounds*: To show the tightness of the upper and lower bounds of the optimal expected value, in Fig. 6, we plot the optimal value (the middle curve) and these bounds (the upper and lower curves) for the static IDs with 3, 5, ..., 13 decision nodes. We see that these bounds are reasonably tight for the tested problems. For example, for $n = 7$, the optimal value is 871.47 while the bounds are 722.91 and 936.637. Although it is difficult to quantitatively analyze the properties of these bounds, these experiments show they can be tight at least for these tested problems.

3) *Computational Gain Under Network-Structure Changes*: To demonstrate how the computational gain of *evalCS* varies with different network structures, we run *evalCS* over the static ID and a modified version of it. The modified ID is obtained as follows: Every link from X_i to Y_j^1 is redirected to Y_j^2 . The resulting ID is shown in the right chart of Fig. 3. Its upstream is $\mathcal{X} \cup Y_{1:m_1}^1 \cup Y_{1:m_2}^2 \cup U_{1:n}$, whereas its downstream is $Y_{1:m_3}^3 \cup \{H\} \cup \{U_H\}$, where $U_{1:n}$ means the set of value nodes, and $Y_{1:m_i}^i$ means the set of nodes Y_j^i , i.e., $Y_{1:m_i}^i = \{Y_1^i, \dots, Y_{m_i}^i\}$ for $i = 1, 2, 3$. Compared with that of

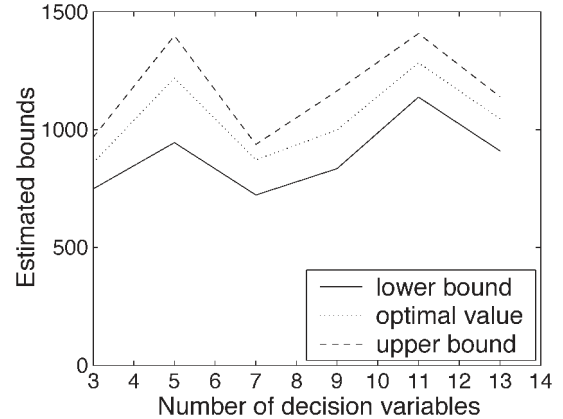


Fig. 6. Lower and upper bounds obtained from the interface utilities.

the static ID, the downstream of the modified ID contains fewer random nodes. We expect: 1) *evalCS* is still more efficient than *evalBF* in the modified ID, since its downstream contains a number of random nodes; and 2) *evalCS* achieves less savings in modified ID than it does in the original ID.

The experiments presented in Fig. 7 confirm these expectations. First, the left chart plots the CPU seconds (in log scale) that *evalCS* and *evalBF* take for the modified IDs with a different number of decision variables. It can be seen that *evalCS* is more efficient. Second, the right chart plots the magnitudes of the savings brought by *evalCS*. For each approach, the saving magnitude is measured by the quotient of the total time of *evalBF* and that of *evalCS*. The magnitudes are drawn in the vertical direction. For a modified ID, *evalCS* is about 14 times faster than *evalBF*. For the original IDs, the magnitudes vary with the number of their decision nodes. We see that the computational savings brought by *evalCS* are more significant for IDs whose downstream contains more nodes.

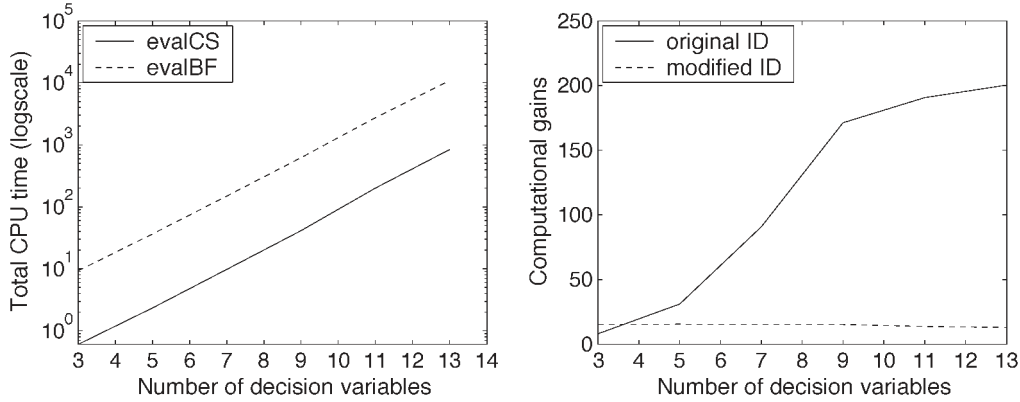


Fig. 7. Computational gains versus network structure.

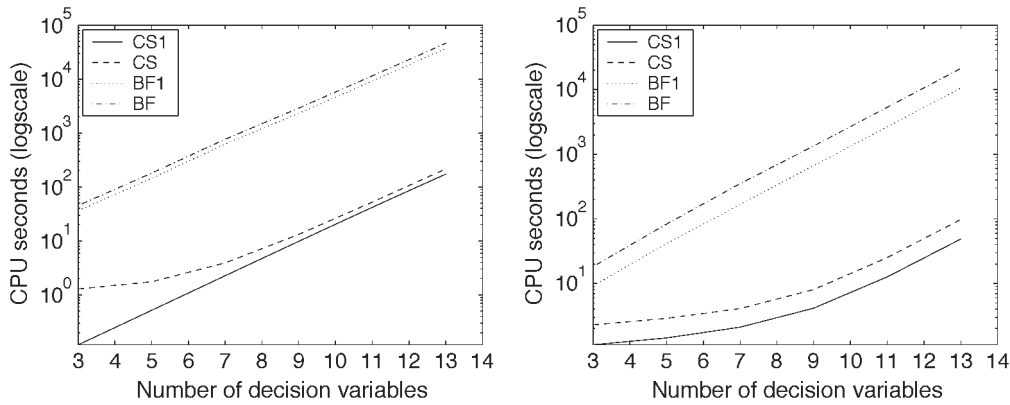


Fig. 8. Replanning for added action/value nodes. In both charts, the curves from the top and bottom plot the total/replanning time for the generic approach, and the total/replanning time for the factorization approach.

4) *Computational Gain Under Network-Parameter Changes:* We also conducted experiments to show how the factorization approach achieves computational savings as the network changes. For this purpose, given the static ID, we first evaluate it (the planning phase), then add more nodes to the ID and reevaluate it (the replanning phase). We like to compare both the replanning time and total time of the factorization approach against that of the generic approach.

In one experiment, we first evaluate a static ID with n decision nodes. We then add two decision nodes to the ID and evaluate the modified ID. Every newly added node has a link from itself to every Y_j^1 node. The timing results in log scale are presented in the left chart of Fig. 8. In the chart, the curve corresponding to CS1 (BF1) depicts the replanning time for the factorization (generic) approach, whereas the curve corresponding to CS (BF) depicts the total time similarly. We see that for the tested problems, the factorization approach achieves considerable savings in replanning when more decision nodes are added. For instance, for the ID with nine decision nodes, the factorization and generic approach, respectively, takes 9.80 and 2313.94 s. These savings are achieved through sharing the interface utilities computed during the evaluation of the original ID. Since the factorization approach takes much less time in both evaluating and reevaluating the ID, its total time is considerably less than that used by the generic approach.

In the other experiment, we evaluate the ID and then add one more value node for replanning. The added value node has a

link from every node Y_j^2 for $j = 1, \dots, 4$. In computing the expected value of the added value node, we still use the factorization approach. In reevaluating an ID, we do not recompute the expected value of the existing value node. The timing results in log scale are collected in the right chart of Fig. 8. The legends read similar to those in the left chart. It can be seen that the factorization approach can achieve great savings in replanning. The reason is obvious: The factorized computations are saved in computing the expected value of the newly added value node. By taking advantage of shared computations in evaluating two value nodes, the total time used by the factorization approach is considerably less than that by the generic approach.

B. A Military Planning Example

We applied the factorization approach to a hypothetical military planning example, which is illustrated in Fig. 9. The overall military goal is to win a war or to bring a tyrant to justice. The goal is represented by a Hypothesis node, which is on the top of the figure. There are 12 primitive actions, namely `destroy_C2`, `destroy_Radars`, ..., `operate_special_force`, which are on the bottom side. Performing an action has direct effects of specific purpose. For instance, if the action `destroy_Radars` is performed, the EW/GCIRadars is destroyed with a high probability. These effects alter the overall goal through altering the low-level subgoals. For instance, the status

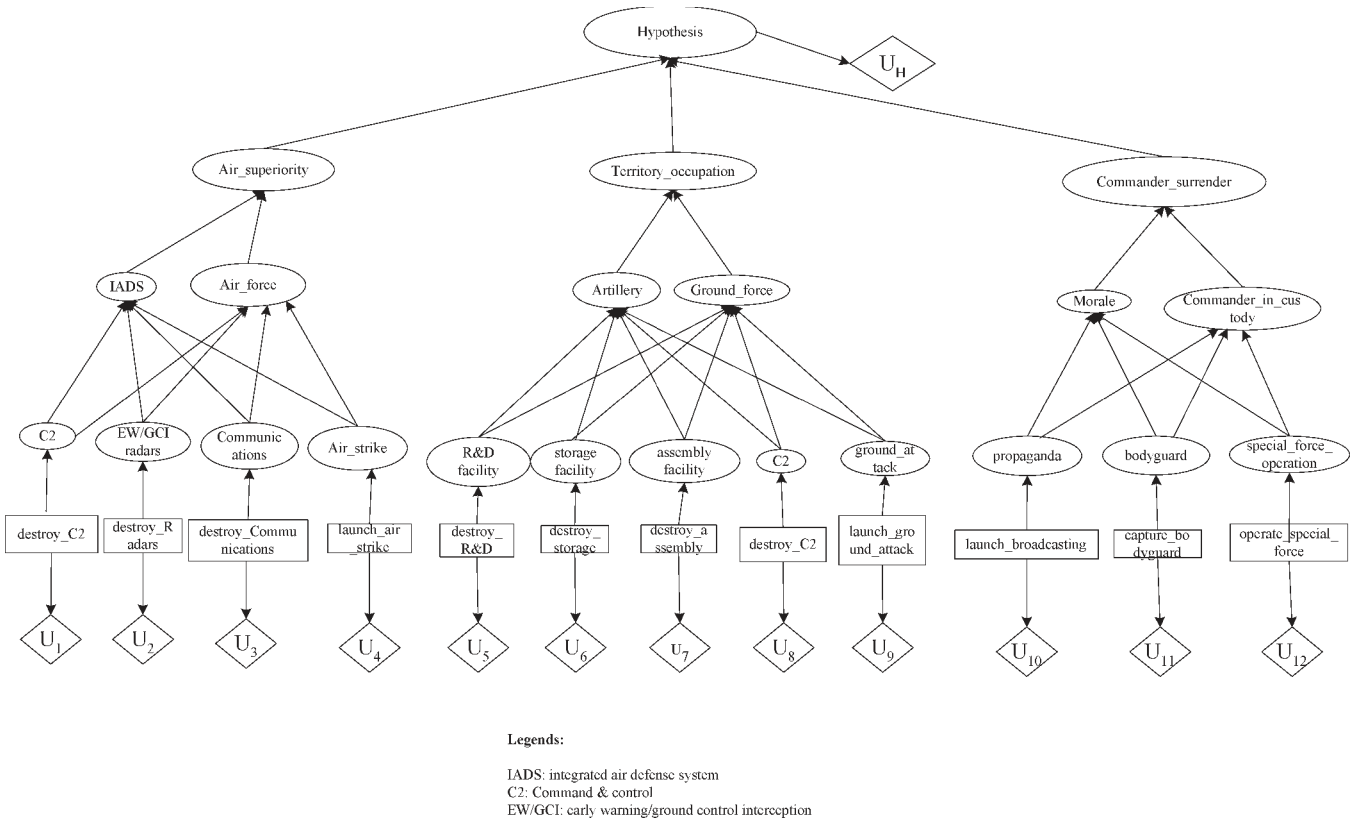


Fig. 9. Static ID illustrating a military planning problem.

of C2 (command and control), EW/GCIRadars and Communications facilities, and Air_strike determine the workability of the integrated air defense system (IADS) and the strength of the enemy air force. In turn, the workability of IADS system and the strength of the enemy air force determine the loss of Air_superiority. The Air_superiority, Territory_occupation, and Commander_surrender are three subgoals determining the overall goal success. Without loss of generality, we assume all nodes are binary. In the example, each decision node is associated with a value node encoding the cost of performing the action, and the hypothesis node is associated with a value node encoding the utility of goal success. The optimal policy needs to balance the utility of goal success and cost of performing actions.

We designed a reasonable set of CPTs and value functions. For the Hypothesis node, if all the subgoals Air_superiority, Territory_occupation, and Command_surrender are achieved, the overall goal is successfully achieved. If one of the subgoals is to be achieved, the probability of the overall success is decreased by 0.3; however, if none of the subgoals is achieved, the overall goal fails with certainty. Similarly, for the subgoal Air_superiority, the two influencing factors are IADS and Air_force. If the IADS system works well and Air_force is strong, Air_superiority is true for the enemy air force; if either the IADS system works poorly or Air_force is weak, the probability of Air_superiority being true is decreased by 0.5. Other CPTs for

Territory_occupation and Command_surrender are set analogously to those for Air_superiority. A similar strategy is used in parameterizing the nodes IADS, Air_force, Artillery, Ground_force, Morale, and Commander_in_custody. In determining the CPTs for random nodes that are immediate children of the decision nodes, we assume that an action achieves its intended effect with probability 0.9. For example, a destroy_Radars decision will destroy the EW/GCI radars with probability 0.9. To complete the ID definition, we also assigned value functions. If the goal is successfully achieved, the reward is 1000; otherwise, the cost, i.e., a negative reward, is 500. For other decision nodes, if a ground attack is launched, the cost is 150; if the special force operation is performed, its cost is 100; if the commander decides to capture the bodyguards of the tyrant, the operating cost is 80; if an air strike is launched, the cost is 50; for any other actions, their operating cost is 20.

Our primary interest is in the performance of the factorization algorithm. From our data, to evaluate the ID, the factorization algorithm took 45 s, while the brute-forced algorithm took 9012 s. Hence, the computational saving is tremendous. We can explain the performance difference by the ID structure—its downstream contains a large number of nodes: all random nodes and the value node associated with the goal. Since its downstream contains far more random nodes than its upstream, the approach is expected to be significantly more efficient. Our secondary interest is concerned with the optimal policy. The optimal policy is the one that performs only air strike and special force operation. The expected value of the ID

is 561.98, and the probability of goal success is 0.81. We note that the optimal policy excludes “launching a ground attack,” although it is the action that is most likely to lead to goal success. One possible reason, as explained earlier, is that the action is excluded due to the high operating cost of performing the action.

VII. CONCLUSION AND FUTURE WORK

In this paper, we studied a special ID class, namely simultaneous IDs, where multiple decisions need to be made at one time step. We intended to make two contributions. First, we examined a simultaneous ID and studied its theoretical properties. We showed that such an ID can be decomposed into an upstream fraction and a downstream fraction, and that the expected value of a value node under a policy can be represented as the sum of interface utilities that involve only the downstream fraction, weighted by the corresponding interface probabilities that involve only the upstream fraction. The interface utilities naturally provide an upper and lower bound of the optimal value of the ID. Second, we proposed a novel factorization algorithm to evaluate a simultaneous ID. The interface utilities are independent of the individual policies; therefore, they can be calculated once but used across all policies in evaluating them. We also extend the factorization approach to a dynamic ID. The algorithm has been tested on simulation studies and a military planning example. Our experiments showed that the factorization algorithm is significantly more efficient than the generic algorithm in evaluating a simultaneous ID.

To further speed up ID evaluating, one future direction is to combine the factorization approach with the approaches of reducing the search space. In this paper, we address one difficulty in ID evaluation, i.e., evaluating individual policies. Another difficulty in ID evaluation is that the policy space contains exponentially many policies and one needs to evaluate all of them in order to find the optimal one. The ID evaluation process can be accelerated if the technique in this paper can be integrated with the approaches of reducing the search space.

ACKNOWLEDGMENT

The authors would like to thank J. Lemmer, D. Gossink, and J. Dussault from AFRL/Rome for introducing the problem to us and for numerous technical exchanges on the related issues. The authors are grateful to three anonymous reviewers for their insightful comments in improving the paper. The authors are also grateful to W. Liao and Z. Zhu for insightful discussions.

REFERENCES

- [1] R. Howard and J. Matheson, “Influence diagrams,” in *The Principles and Applications of Decision Analysis*, R. Howard and J. Matheson, Eds. Menlo Park, CA: Strategic Decisions Group, 1984, pp. 719–762.
- [2] H. Raiffa, *Decision Analysis*. Reading, MA: Addison-Wesley, 1968.
- [3] R. D. Shachter, “Evaluating influence diagrams,” *Oper. Res.*, vol. 34, no. 6, pp. 871–882, 1986.
- [4] P. P. Shenoy, “Valuation-based systems for Bayesian decision analysis,” *Oper. Res.*, vol. 40, no. 3, pp. 463–484, 1992.
- [5] F. Jensen, F. V. Jensen, and S. L. Dittmer, “From influence diagram to junction trees,” in *Proc. 10th Conf. Uncertainty Artificial Intelligence*, Seattle, WA, 1994, pp. 367–373.

- [6] R. Qi and D. Poole, “A new method for influence diagram evaluation,” *Comput. Intell.*, vol. 11, no. 1, pp. 1–34, 1995.
- [7] N. L. Zhang and D. Poole, “Stepwise-decomposable influence diagram,” in *Proc. 3rd Int. Conf. Principles Knowledge Representation and Reasoning*, Cambridge, MA, 1992, pp. 141–152.
- [8] G. F. Cooper, “A method for using belief networks as influence diagrams,” in *Proc. 4th Workshop Uncertainty Artificial Intelligence*, St. Paul, MN, 1988, pp. 55–63.
- [9] R. Shachter and M. Peot, “Decision making using probabilistic inference methods,” in *Proc. 8th Annu. Conf. Uncertainty Artificial Intelligence (UAI)*, Stanford, CA. San Mateo, CA: Morgan Kaufmann, 1992, pp. 276–283.
- [10] T. Nielsen and F. Jensen, “Well-defined decision scenarios,” in *Proc. 15th Conf. Uncertainty Artificial Intelligence*, Stockholm, Sweden, 1999, pp. 502–511.
- [11] F. Jensen and M. Vomlelova, “Unconstrained influence diagrams,” in *Proc. 18th Conf. Uncertainty Artificial Intelligence*, Edmonton, AB, Canada, 2002, pp. 234–241.
- [12] S. L. Lauritzen and D. Nilsson, “Representing and solving decision problems with limited information,” *Manage. Sci.*, vol. 47, no. 9, pp. 1238–1251, 2001.
- [13] N. L. Zhang, “Probabilistic inferences in influence diagrams,” in *Proc. 14th Conf. Uncertainty Artificial Intelligence*, Madison, WI, 1998, pp. 514–522.
- [14] K. J. Aström, “Optimal control of Markov decision processes with incomplete state estimation,” *J. Math. Anal. Appl.*, vol. 10, no. 3, pp. 174–205, 1965.
- [15] U. Kuter, D. Nau, and J. F. Lemmer, “Interactive planning under uncertainty with causal modeling and analysis,” Dept. Comput. Sci., Univ. Maryland, College Park, Tech. Rep. CS-TR-4434, 2003.



Weihong Zhang received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2001.

He then worked as a Postdoc Researcher with the Department of Computer Science and Engineering, Washington University, Saint Louis, MO. He is currently a Postdoc Researcher with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. He has conducted research in artificial intelligence, prob-

abilistic inferences and decision making under uncertainty, graphical models and their applications in sensor networks and human–computer interaction. He has published more than 10 papers in peer-reviewed journals and conferences.



Qiang Ji (S'92–M'98–SM'04) received the Ph.D. degree in electrical engineering from the University of Washington in 1998.

He is currently an Associate Professor with the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute, Troy, NY. His areas of research include computer vision, probabilistic reasoning for decision making and information fusion, pattern recognition, and robotics. He has published more than 70 papers in peer-reviewed journals and conferences. His research has been funded by local and federal government agencies including National Science Foundation (NSF), National Institutes of Health (NIH), Air Force Office of Scientific Research (AFOSR), Office of Naval Research (ONR), Defense Advanced Research Projects Agency (DARPA), and Army Research Office (ARO) and by private companies including Boeing and Honda. His latest research focuses on face detection and recognition, facial-expression analysis, image segmentation, object tracking, user affect modeling and recognition, and active information fusion for decision making under uncertainty.

bilistic inferences and decision making under uncertainty, graphical models and their applications in sensor networks and human–computer interaction. He has published more than 10 papers in peer-reviewed journals and conferences.