Influence Diagrams with Memory States: Representation and Algorithms

Xiaojian Wu, Akshat Kumar, and Shlomo Zilberstein

Computer Science Department
University of Massachusetts
Amherst, MA 01003
{xiaojian,akshat,shlomo}@cs.umass.edu

Abstract. Influence diagrams (IDs) offer a powerful framework for decision making under uncertainty, but their applicability has been hindered by the exponential growth of runtime and memory usage—largely due to the *no-forgetting* assumption. We present a novel way to maintain a limited amount of memory to inform each decision and still obtain near-optimal policies. The approach is based on augmenting the graphical model with *memory states* that represent key aspects of previous observations—a method that has proved useful in POMDP solvers. We also derive an efficient EM-based message-passing algorithm to compute the policy. Experimental results show that this approach produces high-quality approximate polices and offers better scalability than existing methods.

1 Introduction

Influence diagrams (IDs) present a compact graphical representation of decision problems under uncertainty [8]. Since the mid 1980's, numerous algorithms have been proposed to find optimal decision policies for IDs [4,15,9,14,5,11,12]. However, most of these algorithms suffer from limited scalability due to the exponential growth in computation time and memory usage with the input size. The main reason for algorithm intractability is the no-forgetting assumption [15], which states that each decision is conditionally dependent on all previous observations and decisions. This assumption is widely used because it is necessary to guarantee a policy that achieves the highest expected utility. Intuitively, the more information is used for the policy, the better it will be. However, as the number of decision variables increases, the number of possible observations grows exponentially, requiring a prohibitive amount of memory and a large amount of time to compute policies for the final decision variable, which depends on all the previous observations.

This drawback can be overcome by pruning irrelevant and non-informative variables without sacrificing the expected utility [16,17]. However, the analysis necessary to establish irrelevant variables is usually nontrivial. More importantly, this irrelevance or independence analysis is based on the graphical representation of the influence diagram. In some cases the actual probability distribution implies

R.I. Brafman, F. Roberts, and A. Tsoukiàs (Eds.): ADT 2011, LNAI 6992, pp. 306–319, 2011.

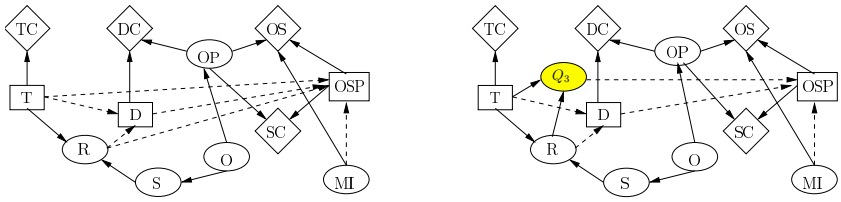

Fig. 1. a) Influence diagram of the oil wildcatter problem (left); b) with a shaded memory node (right). Dotted arrows denote informational arcs.

additional independence relationships among variables that cannot be inferred from the graphical structure. This is usually the case when variables have a large number of successors. Therefore it is beneficial to extract additional (exact or approximate) independence relations in a principled way, thereby decreasing the number of variables that each decision must memorize. In this work, we address this issue by introducing the notion of memory nodes.

Finite-state controllers have been proved very effective in solving infinite-horizon POMDPs [7]. Instead of memorizing long sequences of observations, the idea is to maintain a relatively small number of internal memory states and to choose actions based on this bounded memory. Computing a policy in that case involves determining the action selection function as well as the controller transition function, both of which could be either deterministic or stochastic. With bounded memory, the resulting policy may not be optimal, but with an increasing controller size ϵ-optimality can be guaranteed [2]. A number of search and optimization methods have been used to derive good POMDP policies represented as controllers [1]. More recently, efficient probabilistic inference methods have been proposed as well [19].

Our goal in this paper is to leverage these methods in order to develop more scalable algorithms for the evaluation of IDs. To achieve that, first we introduce a technique to augment IDs with memory nodes. Then, we derive an expectation-maximization (EM) based algorithm for approximate policy iteration for the augmented ID. In the evaluation section, we examine the performance of our algorithm against standard existing techniques.

2 Influence Diagram

An influence diagram (ID) is defined by a directed acyclic graph $G = \{N, A\}$, where N is a set of nodes and A is a set of arcs. The set of nodes, N, is divided into three disjoint groups $\langle X, D, R\rangle$. The set $X = \{X_1, X_2, ..., X_n\}$ is a set of n *chance* nodes, the set $D = \{D_1, D_2, ..., D_m\}$ is a set of m *decision* nodes and $R = \{R_1, R_2, ..., R_T\}$ is a set of T *reward* nodes. Fig. 1(a) shows the influence diagram of the oil wildcatter problem [21], in which decision nodes are illustrated by squares, chance nodes by ellipses and reward nodes by diamonds.

Let $\pi(\cdot)$ and $\Omega(\cdot)$ denote the parents and domain of a node respectively. The domain of a set $Z = \{Z_1, Z_2, ...Z_k\} : Z \subseteq N$, is defined to be the Cartesian

product $\times_{Z_i \in Z} \Omega(Z_i)$ of its individual members' domains. Associated with each chance node is a conditional probability table $P(X_i|\pi(X_i))$. The domain of each decision node is a discrete set of actions. The parents $\pi(D_i)$ of a decision node D_i are called observations, denoted by $O(D_i)$. In other words, decisions are conditioned on the value of their parents [15]. Each reward node R_i defines a *utility function* $g_i(\pi(R_i))$ which maps every joint setting of its parents to a real valued utility.

A stochastic decision rule for a decision node D_i is denoted by δ_i and models the CPT $P(D_i|\pi(D_i); \delta_i) = \delta_i(D_i, \pi(D_i))$. A policy Δ for the ID is a set of decision rules $\{\delta_1, \delta_2, ..., \delta_m\}$, containing one rule for each decision node. Given a complete assignment $\{\boldsymbol{x}, \boldsymbol{d}\}$ of chance nodes X and decision nodes D, the total utility is:

$$U(\boldsymbol{x}, \boldsymbol{d}) = \sum_{i=1}^{T} g_i\big(\{\boldsymbol{x}, \boldsymbol{d}\}_{\pi(R_i)}\big) \tag{1}$$

where $\{\boldsymbol{x}, \boldsymbol{d}\}_{\pi(R_i)}$ is the value of $\pi(R_i)$ assigned according to $\{\boldsymbol{x}, \boldsymbol{d}\}$. The expected utility (EU) of a given policy Δ is equal to

$$\sum_{\boldsymbol{x} \in \Omega(X), \boldsymbol{d} \in \Omega(D)} P\big(\boldsymbol{x}, \boldsymbol{d}\big) U(\boldsymbol{x}, \boldsymbol{d})$$

The probability of a complete assignment $\{\boldsymbol{x}, \boldsymbol{d}\}$ is calculated using the chain rule as follows: $P\big(\boldsymbol{x}, \boldsymbol{d}\big) = \prod_{i=1}^{n} P\big(x_i|\pi(X_i)\big) \prod_{j=1}^{m} \delta_j\big(d_j, \pi(D_j); \Delta\big)$. Therefore, the expected utility is:

$$EU(\Delta; G) = \sum_{\boldsymbol{x} \in \Omega(X), \boldsymbol{d} \in \Omega(D)} \prod_{i=1}^{n} P\big(x_i|\pi(X_i)\big) \prod_{j=1}^{m} \delta_j\big(d_j, \pi(D_j); \Delta\big) U(\boldsymbol{x}, \boldsymbol{d}) \tag{2}$$

The goal is to find the optimal policy $\Delta^\star$ for a given ID that maximizes the expected utility.

A standard ID is typically required to satisfy two constraints [8,15]:

- **Regularity:** The decision nodes are executed sequentially according to some specified total order. In the oil wildcatter problem of Fig. 1(a), the order is $T \prec D \prec OSP$. With this constraint, the ID models the decision making process of a single agent as no decisions can be made concurrently.
- **No-forgetting:** This assumption requires an agent to remember the entire observation and decision history. This implies $\pi(D_i) \subseteq \pi(D_{i+1})$ where $D_i \prec D_{i+1}$. With the no-forgetting assumption, each decision is made based on all the previous information.

3 Influence Diagram with Memory States

The *no-forgetting* assumption makes the policy optimization computationally challenging. In this work, we introduce the notion of influence diagrams with

Algorithm 1. IDMS representation of an influence diagram

input : An ID $G = (N, A)$, k as the number of memory states
1 Create a copy $G_{ms} \leftarrow G$
2 **foreach** *decision node* $i \geq 2$ **do**
3 Add a memory node Q_i to G_{ms} with $|Q_i| = k$
4 Add incoming arcs into Q_i s.t.
5 $\pi(Q_i; G_{ms}) \leftarrow \begin{cases} \pi(D_1; G) & (i = 2) \\ \big(\pi(D_{i-1}; G) \cup Q_{i-1}\big) \backslash \pi(D_{i-2}; G) & (i > 2) \end{cases}$
6 If $\pi(Q_i; G_{ms}) \equiv \phi$, then delete Q_i
7 **foreach** *decision node* $i \geq 2$ **do**
8 **if** $\exists Q_i$ **then**
9 Delete all incoming arcs to D_i in G_{ms}
10 Set the parent of D_i s.t.
11 $\pi(D_i; G_{ms}) \leftarrow \big(Q_i \cup \pi(D_i; G)\big) \backslash \pi(D_{i-1}; G)$
return: the memory bounded ID G_{ms}

memory states (IDMS). The key idea is to approximate the no-forgetting assumption by using limited memory in the form of memory nodes. We start with an intuitive definition and then describe the exact steps to convert an ID into its memory bounded IDMS counterpart.

Definition 1. *Given an influence diagram (ID), the corresponding influence diagram with memory states (IDMS) generated by Alg. 1 approximates the no-forgetting assumption by using new memory states for each decision node, which summarize the past information and provide the basis for current and future decisions.*

The set of memory states for a decision node is represented by a *memory node*. Memory nodes fall into the category of chance nodes in the augmented ID. Such memory nodes have been quite popular in the context of sequential decision making problems, particularly for solving single and multiagent partially observable MDPs [7,13,2]. In these contexts, they are also known as finite-state controllers and are often used to represent policies compactly. Such bounded memory representation provides a flexible framework to easily tradeoff accuracy with the computational complexity of optimizing the policy. In fact, we will show that given sufficient memory states, the optimal policy of an IDMS is equivalent to the optimal policy of the corresponding original ID.

Alg. 1 shows the procedure for converting a given ID, G, into the corresponding memory states based representation G_{ms} using k memory states per memory node. We add one memory node Q_i for each decision node D_i, except for the first decision. The memory nodes are added according to the decision node ordering dictated by the regularity constraint (see line 1). Intuitively, the memory node Q_i summarizes all the information observed up to (not including) the decision node D_{i-1}. Therefore the parents of Q_i include the information summary until the decision D_{i-2} represented by the node Q_{i-1} and the *new* information obtained

after (and including) the decision D_{i-2} and before the decision D_{i-1} (see line 1). Once all such memory nodes are added, we base each decision D_i upon the memory node Q_i and the new information obtained after (and including) the decision D_{i-1} (see line 1). The rest of the incoming arcs to the decision nodes are deleted.

The IDMS approach is quite different from another bounded-memory representation called limited memory influence diagrams (LIMIDs) [11]. A LIMID also approximates the no-forgetting assumption by assuming that each decision depends only upon the variables that can be directly observed while taking the decision. In general, it is quite non-trivial to convert a given ID into LIMID as domain knowledge may be required to decide which information arcs must be deleted and the resulting LIMID representation is not unique. In contrast, our approach requires no domain knowledge and it augments the graph with new nodes. The automatic conversion produces a *unique* IDMS for a given ID using the Alg. 1, parameterized by the number of memory states.

Fig. 1(b) shows an IDMS created by applying Alg. 1 to the ID of the oil wildcatter problem. In the original ID, the order of the decisions is $T \prec D \prec OSP$, namely $D_1 = T$, $D_2 = D$ and $D_3 = OSP$. In the first iteration (see lines 2-6), Q_2 is created as a parent of the node D. However, since T has no parents in the original ID, no parents are added for Q_2 and Q_2 is deleted (see line 6). In the second iteration, Q_3 is created as a parent of OSP, and T, R are linked to Q_3 as its parents because both T and R are parents of D (see line 4 with condition "$i > 2$"). Then, the parents of OSP are reset to be $Q3, D$ and MI (see line 11 with "$i = 3$") because the additional parent of OSP other than D in the original ID is MI.

The CPT of memory nodes, which represents stochastic transitions between memory states, is parameterized by λ: $P\big(Q_i|\pi(Q_i);\lambda_i\big) = \lambda_i(Q_i, \pi(Q_i))$. The decision rules δ for an IDMS are modified according to the new parents. The policy for the IDMS is defined as $\Delta_{ms} = \{\lambda_2, \ldots, \lambda_m, \delta_1, \ldots, \delta_m\}$. The expected utility for an IDMS with policy Δ_{ms}, denoted $EU(\Delta_{ms}; G_{ms})$, is:

$$\sum_{\boldsymbol{x},\boldsymbol{q},\boldsymbol{d}} \prod_{i=1}^{n} P\big(x_i|\pi(X_i)\big) \prod_{j=2}^{m} \lambda_j\big(q_j, \pi(Q_j); \Delta_{ms}\big) \prod_{l=1}^{m} \delta_l\big(d_l, \pi(D_l); \Delta_{ms}\big) U(\boldsymbol{x}, \boldsymbol{d}) \quad (3)$$

The goal is to find an optimal policy $\Delta^{\star}_{ms}$ for the IDMS G_{ms}. As the IDMS approximates the no-forgetting assumption and the value of information is non-negative, it follows that $EU(\Delta^{\star}_{ms}; G_{ms}) \leq EU(\Delta^{\star}; G)$. As stated by the following proposition, an IDMS has far fewer parameters than the corresponding ID. Therefore optimizing the policy for the IDMS will be computationally simpler than for the ID.

Proposition 1. *The number of policy parameters in the IDMS increases quadratically with the number of memory states and remains asymptotically fixed w.r.t. the number of decisions. In contrast, the number of parameters in an ID increases exponentially w.r.t. the number of decisions.*

Proof. The no-forgetting assumption implies that $\pi(D_{i-1}; G) \subseteq \pi(D_i; G)$ in the ID G. Therefore the number of parameters $P(D_i|\pi(D_i); G)$ increases exponentially with the number of decisions. In the IDMS G_{ms}, the size of the parent set of a decision node D_i is $|\pi(D_i; G_{ms})| = |\pi(D_i; G) \backslash \pi(D_{i-1}; G)| + 1$. In many IDs, one can often bound the amount of new information available after each decision by some constant $\mathcal{I} \geq |\pi(D_i; G) \backslash \pi(D_{i-1}; G)|$ for every i. If there are k memory states and the maximum domain size of any node is d, then the number of parameters is $O(d^{\mathcal{I}+1} \cdot k)$ for each decision rule. We can use the same reasoning to show that there are at most $\mathcal{I} + 1$ parents for a controller node Q_i. Therefore the total number of parameters for a controller node is $O(d^{\mathcal{I}} \cdot k^2)$. This shows that overall, parameters increase quadratically w.r.t. the memory states.

Proposition 2. *With a sufficiently large number of memory states, the best policy of an IDMS has the same utility as the best policy of the corresponding ID. Specifically, when* $|\Omega(Q_i; G_{ms})| = |\Omega(\pi(Q_i; G_{ms}))|$ *for all* i, $EU(\Delta^{\star}_{ms}; G_{ms}) = EU(\Delta^{\star}; G)$.

Proof. Let O_i be the set of nodes observed up to (not including) D_i in an IDMS. First, we prove the statement that if in the IDMS, $|\Omega(Q_i)| = |\Omega(\pi(Q_i))|$, then a one-to-one mapping can be built from $\Omega(O_{i-1})$ to $\Omega(Q_i)$. For Q_2, the first memory node, $\pi(Q_2) = O_1$ and the size of Q_2 is equal to $|\Omega(O_1)|$. Thus the mapping can be easily built. Now suppose that the statement is correct for Q_{i-1}. Then for Q_i, since $\pi(Q_i) = \{Q_{i-1}\} \bigcup (O_{i-1}/O_{i-2})$ and a one-to-one mapping from $\Omega(O_{i-2})$ to $\Omega(Q_{i-1})$ already exists, then a one-to-one mapping from O_{i-1} to Q_i can be built similarly in which Q_{i-1} provides all the information of O_{i-2}. Thus, the statement is true for all i. As a result, for each D_i, a one-to-one mapping from O_i to $\pi(D_i; G_{ms})$ can be created such that the no-forgetting condition is satisfied. Therefore, we have $EU(\Delta^{\star}_{ms}; G_{ms}) = EU(\Delta^{\star}; G)$.

4 Approximate Policy Iteration for IDMS

In this section, we present an approximate policy iteration algorithm based on the well known expectation-maximization (EM) framework [6]. The key idea is to transform the policy optimization problem in the IDMS to that of probabilistic inference in an appropriately constructed Bayes net. Such planning-by-inference approach has been shown to be quite successful in Markovian planning problems [20,18,10]; we extend it to influence diagrams. To construct the Bayes net BN_{ms} for a given IDMS, we transform all the reward nodes R_t in the IDMS into binary chance nodes $\hat{R}_t$ with the domain $\Omega(\hat{R}_t) = \{0, 1\}$. The rest of the model is the same as the given IDMS. The CPT of $\hat{R}_t$ is set as follows:

$$P\big(\hat{R}_t = 1|\pi(R_t)\big) \propto g_t(\pi(R_t); G_{ms}) \tag{4}$$

This can be easily done in several ways such as setting $P\big(\hat{R}_t = 1|\pi(R_t)\big) = \big(g_t(\pi(R_t); G_{ms}) - g_{min}\big)/(g_{max} - g_{min})$, where g_{max}, g_{min} denote the maximum and minimum values of the reward.

Proposition 3. *The expected utility of an IDMS is directly proportional to the sum of expectation of binary reward nodes in the corresponding Bayes net:* $EU(\Delta_{ms}; G_{ms}) \propto E\big(\sum_{t=1}^{T} \hat{R}_t\big) + \langle \textit{Ind. terms} \rangle$.

Proof. By the linearity of the expectation, we have:

$$
\begin{aligned}
E\big(\sum_{t=1}^{T} \hat{R}_t; \Delta_{ms}\big) &= \sum_{t=1}^{T} E\big(\hat{R}_t; \Delta_{ms}\big) \qquad (5)\\
&= \sum_{t=1}^{T} P(\hat{R}_t = 1; \Delta_{ms}) \cdot 1 + P(\hat{R}_t = 0; \Delta_{ms}) \cdot 0 \\
&= \sum_{t=1}^{T} \sum_{\pi(R_t)} P(\pi(R_t); \Delta_{ms}) P\big(\hat{R}_t = 1 | \pi(R_t)\big) \\
&= \sum_{t=1}^{T} \sum_{\pi(R_t)} \frac{1}{g_{max} - g_{min}} P(\pi(R_t); \Delta_{ms}) g_t(\pi(R_t); G_{ms}) - \frac{T \cdot g_{min}}{g_{max} - g_{min}} \\
&\propto \sum_{t=1}^{T} \sum_{\pi(R_t)} P(\pi(R_t); \Delta_{ms}) g_t(\pi(R_t)) + \langle \text{Ind. terms} \rangle \\
&= EU(\Delta_{ms}; G_{ms}) + \langle \text{Ind. terms} \rangle \qquad (6)
\end{aligned}
$$

where $\langle$Ind. terms$\rangle$ is a constant with respect to different policies.

4.1 Bayes Net Mixture for IDMS

Intuitively, Proposition 3 and Eq. (5) suggest an obvious method for IDMS policy optimization: if we maximize the likelihood of observing each reward node $\hat{R}_t = 1$, then the IDMS policy will also be optimized. We now formalize this concept using a Bayes net mixture. In this mixture, there is one Bayes net for *each reward node* R_t. This Bayes net is similar to the Bayes net BN_{ms} of the given IDMS, except that it includes only one reward node $\hat{R}$ corresponding to a reward node $\hat{R}_t$ of BN_{ms}; all other binary reward nodes and their incoming arcs are deleted. The parents and the CPT of $\hat{R}$ are the same as that of $\hat{R}_t$. Fig. 2(a) shows this mixture for the oil wildcatter IDMS of Fig. 1(b). The first BN corresponds to the reward node TC, all other reward nodes (DC, OS, SC) are deleted; the second BN is for the node DC. The variable $\mathcal{T}$ is the mixture variable, which can take values from 1 to T, the total number of reward nodes. It has a fixed uniform distribution: $P(\mathcal{T} = i) = 1/T$. The overall approach is based on the following theorem.

Theorem 1. *Maximizing the likelihood* $L(\hat{R}; \Delta_{ms})$ *of observing the variable* $\hat{R} = 1$ *in the Bayes net mixture (Fig. 2(a)) is equivalent to optimizing the IDMS policy.*

Proof. The likelihood for each individual BN in the BN mixture is $L_t^{\Delta_{ms}} = P(\hat{R} = 1 | \mathcal{T}; \Delta_{ms})$, which is equivalent to $P(\hat{R}_t = 1; \Delta_{ms})$ in the Bayes net

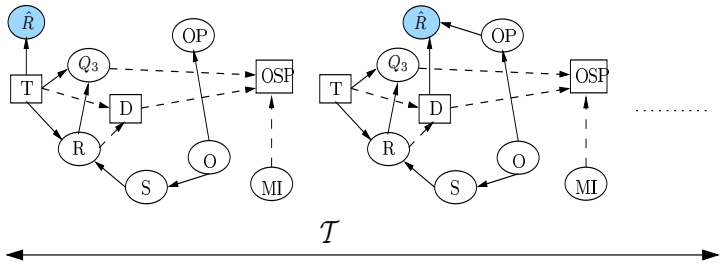

Fig. 2. Bayes net mixture for the oil wildcatter problem

BN_{ms}. Note that the deleted binary reward nodes in each individual BN of the mixture do not affect this probability. Therefore the likelihood for the complete mixture is:

$$L(\hat{R}; \Delta_{ms}) = \sum_{t=1}^{T} P(\mathcal{T} = t) L_t^{\Delta_{ms}} = \frac{1}{T} \sum_{t=1}^{T} P(\hat{R}_t = 1; \Delta_{ms}) \tag{7}$$

From Proposition 3, we now have $L(\hat{R}; \Delta_{ms}) \propto EU(\Delta_{ms}; G_{ms})$. Therefore maximizing the likelihood for the mixture would optimize the IDMS policy.

Note that for the implementation, we do not explicitly create the mixture; all the computations on this mixture can be directly performed on the single Bayes net BN_{ms}.

4.2 The Expectation Maximization (EM) Algorithm

We now derive the E and M-step of the expectation-maximzation framework that can be used to maximize the above likelihood [6]. In the EM framework, the observed data is $\hat{R} = 1$; the rest of the variables are hidden. The parameters to optimize are the policy parameters for the IDMS: the λ's for the memory nodes and δ's for the decision nodes. The full joint $P(\hat{R}, X, D, Q, \mathcal{T}; \Delta_{ms})$ for the BN mixture is given by:

$$P\big(\hat{R}|\pi(\hat{R}), \mathcal{T}\big) \prod_{i=1}^{n} P(X_i|\pi(X_i)) \prod_{j=1}^{m} \delta_j\big(D_j, \pi(D_j)\big) \prod_{l=2}^{m} \lambda_l\big(Q_l, \pi(Q_l)\big) \tag{8}$$

We will omit specifying Δ_{ms} as long as it is unambiguous. As EM maximizes the log-likelihood, we take the log of the above to get:

$$\log P(\hat{R}, X, D, Q, \mathcal{T}; \Delta_{ms}) = \sum_{j=1}^{m} \delta_j\big(D_j, \pi(D_j)\big) + \sum_{l=2}^{m} \lambda_l\big(Q_l, \pi(Q_l)\big) + \langle \text{Ind. terms} \rangle \tag{9}$$

where ⟨Ind. terms⟩ denote terms independent of the parameters λ and δ. EM maximizes the expected log-likelihood $Q(\Delta_{ms}, \Delta^{\star}_{ms})$ to be equal to:

$$\sum_{\mathcal{T}=1}^{T} \sum_{X,D,Q} P(\hat{R}=1, X, D, Q, \mathcal{T}; \Delta_{ms}) \log P(\hat{R}=1, X, D, Q, \mathcal{T}; \Delta^{\star}_{ms}) \qquad (10)$$

where Δ_{ms} is the current policy and $\Delta^{\star}_{ms}$ is the policy to be computed for the next iteration. We first show the update rule for decision node parameters δ.

$$\begin{aligned} Q(\Delta_{ms}, \Delta^{\star}_{ms}) &= \sum_{\mathcal{T}=1}^{T} \sum_{X,D,Q} P(\hat{R}=1, X, D, Q, \mathcal{T}; \Delta_{ms}) \sum_{j=1}^{m} \log \delta_j\big(D_j, \pi(D_j); \Delta^{\star}_{ms}\big) \\ &= \sum_{j=1}^{m} 1/T \sum_{D_j, \pi(D_j)} \sum_{\mathcal{T}=1}^{T} P(\hat{R}=1, D_j, \pi(D_j) | \mathcal{T}; \Delta_{ms}) \log \delta_j\big(D_j, \pi(D_j); \Delta^{\star}_{ms}\big) \end{aligned}$$

The above expression can be easily maximized for each parameter δ_j using the Lagrange multiplier for the normalization constraint:

$$\forall \pi(D_j) : \sum_{D_j} \delta_j\big(D_j | \pi(D_j)\big) = 1.$$

The final updated policy is:

$$\delta_j\big(D_j, \pi(D_j); \Delta^{\star}_{ms}\big) = \frac{\sum_{\mathcal{T}=1}^{T} P(\hat{R}=1, D_j, \pi(D_j) | \mathcal{T}; \Delta_{ms})}{C_{\pi(D_j)}} \qquad (11)$$

where $C_{\pi(D_j)}$ is the normalization constant. The memory node parameter (λ) update equation is analogous to the above with the node D_i replaced by Q_l. The above equation describes the M-step. We next describe the E-step that involves computing the probabilities $P(\hat{R}=1, (\cdot), \pi(\cdot) | \mathcal{T}; \Delta_{ms})$ where $(\cdot)$ ranges over the decision and memory nodes.

4.3 Probabilities Computation

The join-tree algorithm is an efficient algorithm for computing marginal probabilities [3]. The algorithm performs inference on the Bayesian network by transforming it into a join-tree. The tree satisfies the running intersection property. Each tree node represents a clique containing a set of nodes in the BN_{ms}. An advantage of this algorithm is that any node and its parents are included in at least one clique. Therefore by performing a global message passing, the joint probabilities of nodes and its parents with a given evidence can be obtained from cliques implementing the E-step.

Alg. 2 describes the procedure to update the decision rules $\delta_i\big(D_i, \pi(D_i)\big)$. In each iteration, one of the variables R_t is set to 1 and the corresponding probabilities are calculated. New parameters are computed using Eq. (11).

Algorithm 2. Procedure for updating $\delta_j\big(D_j, \pi(D_j)\big)$

input : BN_{ms} – the transformed Bayesian network
Build the join-tree for BN_{ms}
Initialize parameters δ_i randomly $\forall i = 1 : m$
repeat
  Initialize $V\big(D_i, \pi(D_i)\big) \leftarrow 0$
  for $t = 1 : T$ **do**
    Set evidence $R_t = 1$ to every clique containing R_t
    Conduct a global message passing on the join-tree
    Compute $P(R_t = 1, D_i, \pi(D_i))$ by marginalization $\forall i = 1 : m$
    $V(D_i, \pi(D_i)) \leftarrow V(D_i, \pi(D_i)) + P(R_t = 1, D_i, \pi(D_i))$
    Recover potentials and clear evidence
  $\delta_i^{\star}\big(D_i, \pi(D_i)\big) = V(D_i, \pi(D_i))/C_{\pi(D_i)}$ (C $\equiv$ normalization constant)
  Set $\delta_i^{\star}$ into BN_{ms}
until *the convergence criterion is satisfied*
return: the BN_{ms} with updated policy parameters

Fig. 3 shows the join-tree of the oil wildcatter problem. The performance of Alg. 2 is mainly determined by the size of the largest clique or tree-width of the join-tree. The size of the cliques is influenced largely by the number of parents of each node because each node and its parent are contained in at least one clique (family preserving property). Therefore this algorithm will be more efficient for the IDMS as the number of parents of each node is much smaller that in the ID.

5 Experiments

In this section, we compare the EM algorithm against Cooper's algorithm [4], implemented in SMILE, a library created by the Decision Systems Lab at U. Pitt. We test the algorithms on two datasets: randomly generated IDs and Bayesian networks converted into IDs. The Cooper's algorithm provides optimal solutions.

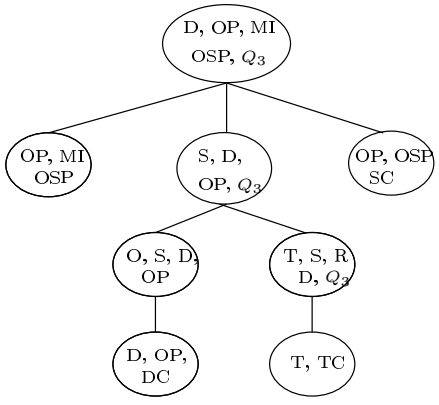

Fig. 3. Join Tree of Oil wildcatter problem

5.1 Randomly Generated IDs

We randomly generated IDs with different settings and fixed the number of parents of chance nodes and reward nodes to be 2. Each decision node has two more parents than the previous decision node (the no-forgetting assumption is forced). With 0.1 probability, a chance node degenerates into a deterministic

Table 1. 'C40' and 'C60' denote the number of chance nodes (40 and 60 respectively). All the networks have 6 reward nodes. 'D' is the number of decision nodes. '-' means that Cooper's algorithm ran out of memory before terminating. T denotes time in seconds. M denotes memory required in MB. Loss is equal to $\big(EU(Cooper) - EU(EM)\big)/EU(Cooper)$.

C40	Cooper		EM			C60	Coopers		EM		
D	T	M	T	M	Loss	D	T	M	T	M	Loss
4	1.1	5.3	0.2	7.0	<1.0%	4	1.2	5.3	0.8	7.0	<1.0%
5	7.2	8.1	0.2	8.0	1.2%	5	7.1	8.1	1.6	8.0	<1.0%
6	24.2	11.5	0.4	10.7	1.0%	6	25.4	12.0	5.6	10.7	<1.0%
7	106.7	48.6	0.6	16.5	<1.0%	7	112.6	48.2	1.1	16.5	<1.0%
8	264.0	227.0	1.3	31.1	1.6%	8	256.8	227.0	6.8	31.1	<1.0%
9	-	>764	2.4	111.0	-	9	-	>763.8	2.7	111.0	-
10	-	-	3.1	111.0	-	10	-	-	2.0	111.0	-
11	-	-	5.1	150.0	-	11	-	-	16.7	150.0	-
12	-	-	6.7	207.0	-	12	-	-	18.9	207.0	-
13	-	-	5.6	207.0	-	13	-	-	37.2	207.0	-

node. In order to increase bias, for each reward node, the reward value is in range $[0, 20]$ with 40% probability, in [20,70] with 20% probability and in $[70, 100]$ with 40% probability. For each network setting, 10 instances are tested and the average is reported. The results are shown in Table 1. In these experiments, Cooper's algorithm ran on the original ID (no-forgetting) and EM on an IDMS with 2 states per memory node. As the number of decision nodes increases, the running time and memory usage of Cooper 's algorithm grows much faster than EM's. When the ID has 9 decision nodes, Cooper's algorithm fails to terminate, but EM can still solve the problem in less than 3 seconds using only 111 MB of memory. Furthermore, EM provides good solution quality. The value loss against Cooper's algorithm (which is optimal) is about 1%.

5.2 Bayesian Networks Transformed into IDs

Since real world decision problems are likely to have more structure and nodes are usually not randomly connected, we also experimented with the Bayesian network samples available on the GENIE website. We built IDs by transforming a portion of chance nodes into decision nodes and also adding a certain number of reward nodes. Two Bayesian network datasets were used. The average results are reported in Table 2. In both of these benchmarks, EM again performs much better w.r.t. runtime and the solution quality loss remains small, around 1%. On these benchmarks, both EM and Cooper's algorithm are faster than on the random graphs as many of these Bayes nets are tree-structured.

5.3 The Effect of Memory States

In this section, we examine how well memory states approximate the no-forgetting assumption and the effect of the number of memory states on the overall quality

Table 2. Results for the Hepar II and Win95pts Bayesian network datasets. D, C, R represent the number of decision nodes, chance nodes and reward nodes respectively. T is the running time in seconds. M is the amount of memory in MB.

Hepar II			Coopers		EM			Win95pts			Coopers		EM		
D	C	R	T	M	T	M	Loss	D	C	R	T	M	T	M	Loss
14	61	5	47.27	759.8	0.22	3.5	1.5%	13	63	5	45.05	759.8	0.26	3.4	1.5%
15	60	5	15.17	760.1	0.21	3.7	1.1%	14	62	5	14.81	760.7	0.23	3.6	$< 1\%$
16	59	5	10.98	760.3	0.26	3.7	$< 1\%$	15	61	5	10.66	761.1	0.21	3.6	$< 1\%$
17	58	5	22.02	761.7	0.24	4.0	$< 1\%$	16	60	5	21.29	762.6	0.22	3.9	$< 1\%$
18	57	5	14.20	762.3	0.21	4.3	$< 1\%$	17	59	5	13.86	763.2	0.22	4.3	$< 1\%$
18	57	5	15.35	762.6	0.22	4.6	$< 1\%$	18	58	5	14.71	763.4	0.21	4.6	$< 1\%$

achieved by EM. For simplicity, we use a small ID containing only three nodes: a chance node, a decision node, and a reward node as their child. We let both the chance node and the decision node have 50 states and the value of the chance node is distributed uniformly. This simple ID can be easily made to represent more complex situations. For example, we can replace the chance node with a complex Bayes net and similarly replace the reward node by a Bayes net with additional reward nodes.

In this simple ID, we assume that the chance node models some events that occurred much earlier such that the current decision node does not observe them directly. However, the nodes have some effect on the reward obtained, so a memory node is provided that could record the value of the chance node so that the right decision can be made. I would like to change as: In order to test the effect of increasing the size of the memory node on the expected utility, we assign value for the reward node such that for each value of the chance node, only one action (selected randomly) of the decision node produces the reward "1" and all the other actions produce "0". In this way, it is crucial to know the value of the chance node in order to maximize the expected utility.

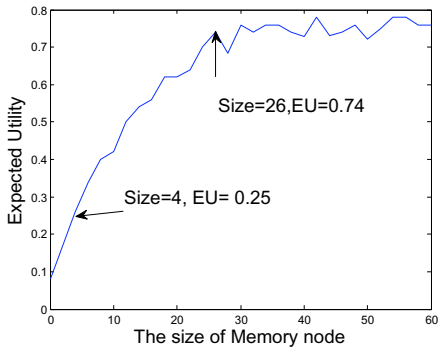

Fig. 4. The effects of memory states w.r.t. expected utility

When the size of the memory node is 50, then according to Proposition 2, the maximum expected utility that can be obtained by an *optimal* policy is 1. In these experiments, we tested the EM algorithm with different sizes of the memory node. The results, shown in Fig. 4, confirm that the EU increases quickly at the beginning and then remains almost constant at about 26 memory states.

Note that the EU does not reach 1 with 50 memory states because the EM algorithm converges to local optima. This example illustrates a case in which a large size of the memory node is needed in order to obtain good solutions. We also note that this experiment is deliberately designed to test the impact of violating the no-forgetting assumption in an extreme situation. In practice, we anticipate that smaller memory nodes will suffice because reward nodes are not as tightly coupled with chance nodes as in these experiments.

6 Conclusion

In this paper, we introduce a technique to transform an influence diagram into an influence diagram with memory states by relaxing the no-forgetting assumption. We also develop the EM algorithm to solve the resulting IDMS efficiently. We show that there exist problems that require large memory states to obtain good quality solutions. However, experiments with both randomly generated and standard benchmark IDs yield near-optimal policies using a small number of memory states. This work unifies techniques for solving (decentralized) POMDPs using finite-state controllers and solving large influence diagrams. The connections we establish in this work between various memory-bounded approximations will facilitate greater sharing of results between researchers working on these problems.

Acknowledgements. This work was funded in part by the National Science Foundation under grant IIS-0812149 and the Air Force Office of Scientific Research under grant FA9550-08-1-0181.

References

1. Amato, C., Bernstein, D.S., Zilberstein, S.: Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. Autonomous Agents and Multi-Agent Systems 21, 293–320 (2010)
2. Bernstein, D.S., Amato, C., Hansen, E.A., Zilberstein, S.: Policy iteration for decentralized control of Markov decision processes. Journal of Artificial Intelligence Research 34, 89–132 (2009)
3. Cecil Huang, A.D.: Inference in belief networks: A procedural guide. International Journal of Approximate Reasoning 15, 225–263 (1994)
4. Cooper, G.: A method for using belief networks as influence diagrams. In: Proc. of the Conference on Uncertainty in Artificial Intelligence, pp. 55–63 (1988)
5. Dechter, R.: A new perspective on algorithims for optimizing policies under uncertainty. In: Proc. of the International Conference on Artificial Intelligence Planning Systems, pp. 72–81 (2000)
6. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical society, Series B 39(1), 1–38 (1977)
7. Hansen, E.A.: An improved policy iteration algorithm for partially observable MDPs. In: Proc. of Neural Information processing Systems, pp. 1015–1021 (1997)

8. Howard, R.A., Matheson, J.E.: Infuence diagrams. In: Readings on the Principles and Applications of Decision Analysis, vol. II, pp. 719–762. Strategic Decisions Group (1984)
9. Jensen, F., Jensen, F.V., Dittmer, S.L.: From influence diagrams to junction trees. In: Proc. of the Conference on Uncertainty in Artificial Intelligence, pp. 367–373 (1994)
10. Kumar, A., Zilberstein, S.: Anytime planning for decentralized POMDPs using expectation maximization. In: Proc. of the Conference on Uncertainty in Artificial Intelligence, pp. 294–301 (2010)
11. Nilsson, D., Lauritzen, S.: Representing and solving decision problems with limited information. Management Science 47(9), 1235–1251 (2001)
12. Marinescu, R.: A new approach to influence diagram evaluation. In: Proc. of the 29th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (2009)
13. Poupart, P., Boutilier, C.: Bounded finite state controllers. In: Proc. of Neural Information processing Systems, pp. 823–830 (2003)
14. Qi, R., Poole, D.: A new method for influence diagram evaluation. Computational Intelligence 11, 498–528 (1995)
15. Shachter, R.: Evaluating influence diagrams. Operations Research 34, 871–882 (1986)
16. Shachter, R.: Probabilistic inference and influence diagrams. Operations Research 36, 589–605 (1988)
17. Shachter, R.: An ordered examination of influence diagrams. Networks 20, 535–563 (1990)
18. Toussaint, M., Charlin, L., Poupart, P.: Hierarchical POMDP controller optimization by likelihood maximization. In: Proc. of the Conference on Uncertainty in Artificial Intelligence, pp. 562–570 (2008)
19. Toussaint, M., Harmeling, S., Storkey, A.: Probabilistic inference for solving (PO)MDPs. Technical Report EDI-INF-RR-0934, School of Informatics, University of Edinburgh (2006)
20. Toussaint, M., Storkey, A.J.: Probabilistic inference for solving discrete and continuous state Markov decision processes. In: Proc. of International Conference on Machine Learning, pp. 945–952 (2006)
21. Zhang, N.L., Qi, R., Poole, D.: A computational theory of decision networks. International Journal of Approximate Reasoning 11, 83–158 (1994)