Lifted Stochastic Planning, Belief Propagation and Marginal MAP

Hao Cui, Roni Khardon
Department of Computer Science, Tufts University, Medford, Massachusetts, USA
Hao.Cui@tufts.edu, roni@cs.tufts.edu

Abstract

It is well known that the problems of stochastic planning and probabilistic inference are closely related. This paper makes several contributions in this context for factored spaces where the complexity of solutions is challenging. First, we analyze the recently developed SOGBOFA heuristic, which performs stochastic planning by building an explicit computation graph capturing an approximate aggregate simulation of the dynamics. It is shown that the values computed by this algorithm are identical to the approximation provided by Belief Propagation (BP). Second, as a consequence of this observation, we show how ideas on lifted BP can be used to develop a lifted version of SOGBOFA. Unlike implementations of lifted BP, Lifted SOGBOFA has a very simple implementation as a dynamic programming version of the original graph construction. Third, we show that the idea of graph construction for aggregate simulation can be used to solve marginal MAP (MMAP) problems in Bayesian networks, where MAP variables are constrained to be at roots of the network. This yields a novel algorithm for MMAP for this subclass. An experimental evaluation illustrates the advantage of Lifted SOGBOFA for planning.

Introduction

The connection between planning and inference is well known both for the logical setting and the probabilistic setting. Over the last decade multiple authors have introduced explicit reductions showing how stochastic planning can be solved using probabilistic inference (Domshlak and Hoffmann 2006; Toussaint and Storsky 2006; Furmston and Barber 2010; Lang and Toussaint 2010; Furmston and Barber 2011; Neumann 2011; Kober and Peters 2011; Sabbadin, Peyrard, and Forsell 2012; Liu and Ihler 2012; Cheng et al. 2013; Kiselev and Poupart 2014; Nitti, Belle, and Raedt 2015; van de Meent et al. 2016; Lee, Marinescu, and Dechter 2016; Nguyen, Kumar, and Lau 2017) with applications in robotics, scheduling and environmental problems. However, heuristic methods and search are still the best performing approaches for planning in large combinatorial state and action spaces (Kolobov et al. 2012; Keller and Helmert 2013; Cui and Khardon 2016). This paper makes several contributions in this context.

We analyze a recent heuristic algorithm that was shown to be very effective in practice for large action spaces. SOGBOFA (Cui and Khardon 2016) builds an approximate computation graph capturing marginals of state and reward variables under independence assumption. It then uses gradient based search to optimize action choice. Our analysis shows that the value computed by SOGBOFA’s computation graph is identical to the solution of Belief Propagation (BP) when conditioned on actions. Abstracting this suggests a novel approach to planning and marginal MAP problems, where approximations are often used: If we can compute a closed form representation of the value (or marginal probability) computed by the chosen approximation, when conditioned on the actions (max variables), then we can perform the optimization through automatic differentiation (Griewank and Walther 2008). In the case of SOGBOFA we have a closed form for the result of BP for a sub-family of directed graphs. This captures marginal MAP (MMAP) problems for Bayesian networks with max variables at roots and evidence restricted to be at one leaf node.

Given this analysis it is natural to ask if we can improve SOGBOFA further by taking advantage of improvements of BP, for example Lifted BP (Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009) where repeated messages are avoided in the computation. As we show, this can be done, and the form of the lifted algorithm is extremely simple. For the sub-family of directed graphs, lifted BP is achieved simply by using a dynamic programming implementation of the construction of the SOGBOFA graph. This is an important advantage because typical implementations of lifted inference are complex, whereas here the implementation comes almost for free. An experimental evaluation shows that the computation time of SOGBOFA is improved and that, under the same time constraints, lifted SOGBOFA improves the performance in planning over SOGBOFA.

Finally, while SOGBOFA was developed for planning, it can be used to solve MMAP problems. The original construction for SOGBOFA assumes one final $Q$ value node which is being optimized. It is not hard to see that this can be adapted to solve any MMAP problem with decision variables at the roots of the Bayesian network. This provides a novel alternative to the mixed-BP algorithm of (Liu and Ihler 2013) for MMAP, where our algorithm combines lifted BP conditioned on MAP nodes with gradient search.
Preliminaries

We assume familiarity with basic notions in Bayesian networks and stochastic planning. This section recalls some details of BP and SOGOBOFA.

Belief Propagation in Bayesian Networks

For our results it is convenient to refer to the BP algorithm for directed graphs (Pearl 1989). A Bayesian network is given by a directed acyclic graph where each node is associated with a random variable and a corresponding conditional probability table (CPT) capturing \( p(x|\text{parents}(x)) \). The joint probability of all variables is given by \( \prod_{i} p(x_{i}|\text{parents}(x_{i})) \).

We restrict the presentation in this paper to Bayesian networks with binary variables, i.e., \( x_{i} \in \{0, 1\} \). BP calculates an approximation of the marginal probability of each node by sending messages among adjacent nodes.

Assume first that the directed graph is a polytree (no underlying undirected cycles). The marginal of \( x \) has two values. The algorithm is initialized as follows. If \( x \) is a root (parent-less), we set \( \lambda(x) \equiv 0 \) and \( \pi(x) \equiv p(e|x) \), where \( e \) is the evidence, i.e., \( x_{i} \in \{0, 1\} \). BP calculates an approximation of the marginal probability of each node by sending messages among adjacent nodes.

The messages are given by

\[
\lambda_{x}(w_{i}) = \beta \sum_{w_{i}} p(x|w_{i}) \prod_{k \neq i} \pi_{x}(w_{k})
\]

\[\lambda(x) = \prod_{j=1}^{n} \lambda_{z_{j}}(x) \]

(2)

and

\[
\pi(x) = \sum_{w} p(x|w) \prod_{k=1}^{n} \pi_{x}(w_{k})
\]

In on-line planning we are given a fixed limited time \( t \) per step and cannot compute a policy in advance. Instead, given the current state, the algorithm must decide on the next action consuming at most \( t \) time. Then the action is performed in the MDP, a transition and reward are observed and the algorithm is presented with the next state. This process repeats and the long term performance of the algorithm is evaluated. On-line planning has been the standard evaluation method in recent planning competitions. The algorithms discussed in this paper perform on-line planning by estimating the value of every initial action where a rollout policy, typically a random policy, is used in future steps. They differ in how they estimate the value of an initial action through a forward simulation and in how they search over possibilities for the initial action.

The AROLLOUT algorithm (Cui et al. 2015) introduced the idea of algebraic simulation to estimate values but optimized over actions in a brute force manner. The work of (Cui and Khardon 2016) showed how rollouts can be computed symbolically and that the optimization can be done using automatic differentiation leading to the SOGOBOFA algorithm.
We first show that the computation of \( \text{AROLLOUT} \) can be rewritten as a sum over assignments. The sum in 8 can be divided into disjoint sets of assignments to the parents of \( c_{ij} \). In other words, the CPT is represented in the form if \( c_{i1} \mid c_{i2} \ldots \) then \( p_1 \) \ldots if \( c_{n1} \mid c_{n2} \ldots \) then \( p_n \), and the elements \( c_{ij1} \) and \( c_{ij2} \) are mutually exclusive and exhaustive. \( \text{AROLLOUT} \) performs a forward pass calculating an approximation of the true marginal \( p(x) \), for any node in the graph. \( p(x) \) is calculated as a function of \( \hat{p}(c_{ij}) \), an estimate of the probability that \( c_{ij} \) is true.

\[
\hat{p}(x) = \sum_{ij} p(x|c_{ij}) \hat{p}(c_{ij}) = \sum_{ij} p_i \hat{p}(c_{ij}) \tag{6}
\]

where \( \hat{p}(c_{ij}) \) is calculated assuming its parents are independent.

\[
\hat{p}(c_{ij}) = \prod_{w_k \in c_{ij}} \hat{p}(w_k) \prod_{\bar{w}_k \in c_{ij}} (1 - \hat{p}(w_k)). \tag{7}
\]

The main observation in SOGBOTA is that instead of using this process to calculate concrete values, we can use the expressions to construct an explicit data structure representing the computation graph, where the last node represents the accumulated reward. Once this is done, we can use automatic differentiation and gradient based search to optimize action variables. SOGBOTA includes several additional heuristics including dynamic control of simulation depth, dynamic selection of gradient step size, maintaining domain constraints, and a balance between gradient search and random restarts. Most of this is orthogonal to topic of this paper and we omit the details. We discuss the control of simulation depth with the experimental section, where it bears on the results presented.

**AROLLOUT is Equivalent to BP**

We first show that the computation of \( \text{AROLLOUT} \) can be rewritten as a sum over assignments.

**Lemma 2.** \( \text{AROLLOUT} \)'s calculation in Eq (6) and (7) is equivalent to

\[
\hat{p}(x) = \sum_{W} p(x|W) \prod_{w_k \in c_{ij}} \hat{p}(w_k) w_k (1 - \hat{p}(w_k))^{1-w_k} \tag{8}
\]

where \( W \) is the set of assignment to the parents of \( x \).

**Proof.** The sum in 8 can be divided into disjoint sets of assignments according to the \( c_{ij} \) they satisfy. Let \( w_l \) be a parent of \( x \) which is not in \( c_{ij} \). Let \( W(c_{ij}) \) be the assignments to the parents of \( x \) which satisfy \( c_{ij} \), and \( W_{\backslash w_l}(c_{ij}) \) be the assignments to the parents of \( x \) except for \( w_l \) which satisfy \( c_{ij} \). Since \( w_l \) is not in \( c_{ij} \), \( W_{\backslash w_l}(c_{ij}) \) is well defined. We have

\[
\sum_{W(c_{ij})} p(x|W) \prod_{w_k \in c_{ij}} \hat{p}(w_k) w_k (1 - \hat{p}(w_k))^{1-w_k} \tag{9}
\]

where the last equality is true because \( \pi(x) \) is always normalized. Therefore from Eq (3) we have

\[
BEL(x) = \sum_{w} p(x|w) \prod_{k=1}^{n} \pi_x(w_k). \tag{11}
\]

Now, from Eq (5) and Lemma 1 we have

\[
\pi_x(w_k) = \pi(w_k) = BEL(w_k) \tag{12}
\]
and substituting (12) into (11) we get

\[ \text{BEL}(x) = \sum_{w} p(x|w) \prod_{k=1}^{n} \text{BEL}(w_k). \quad (13) \]

Inductively assuming \( \text{BEL}(w_k = T) = \hat{p}(w_k) \) and \( \text{BEL}(w_k = F) = 1 - \hat{p}(w_k) \), we can rewrite Eq (13) as \( \text{BEL}(x = T) = \hat{p}(x) = \sum_{w} p(x|W) \prod_{k=1}^{n} \hat{p}(w_k) (1 - \hat{p}(w_k))^{1-w_k} \), which is identical to Eq (8). \[ \square \]

**Lifted SOGBOFA**

Given the equivalence of the previous section, we next show how SOGBOFA can be improved using ideas from lifted inference. Lifted BP aims to calculate exactly the same solutions as BP but do so more quickly by avoiding identical messages. The idea was first introduced by (Singla and Domingos 2008) who showed how to extract the equivalence classes of messages from a structured model, specifically a Markov logic network. Each type of message is then computed only once and propagated as a power of the original individual message, replacing the product of identical messages. The work of (Kersting, Ahmadi, and Natarajan 2009) showed how repeated structure can be extracted from any ground Markov network by simulating only the process of BP without calculating the messages. The messages are then grouped and propagated similarly.

The details of Lifted BP were developed for undirected models but the same principles apply to directed models. Thanks to the property of BP exposed in Lemma 1 we know that in SOGBOFA graphs there are no backward messages, and we can focus on the structure of forward messages. Further, forward messages are the marginal distributions of the corresponding variables, and since we are focusing on binary variables each message can be captured with one number or expression for \( p(x_1 = 1) \). In Lifted BP, two nodes or factors send identical messages when all their input messages are identical and in addition the local factor is identical. With the forward structure of SOGBOFA this has a clear analogy. Two nodes in the SOGBOFA graph are identical when their parents are identical and the local algebraic expressions capturing the local factors are identical. This suggests a straightforward implementation for Lifted SOGBOFA using dynamic programming.

**Lifted SOGBOFA algorithm:** We run the algorithm in exactly the same manner as SOGBOFA except for the construction of the computation graph. (1) When constructing a node in the explicit computation graph we check if an identical node, with same parents and same operation, has been constructed before. If so we return that node instead of constructing a new one. Otherwise we create a new node in the graph. (2) If we only use the idea above we may end up with multiple edges between the same two nodes. Such structures are simplified on the fly during construction. In particular, every plus node with \( k \) incoming identical edges is turned into a multiply node with constant multiplier \( k \). Similarly, every multiply node with \( k \) incoming identical edges is turned into a power node with constant multiplier \( k \). This portion of the construction generates the counting expressions, often seen in lifted inference, automatically from the DBN structure.

It is important to identify two aspects of this algorithm and its relation to lifted BP. First, similar to (Kersting, Ahmadi, and Natarajan 2009) and unlike (Singla and Domingos 2008), our algorithm for constructing the lifted graph must first process the ground network, i.e., its complexity is linear in the size of the original ground computation graph. It is interesting to explore constructions that go directly from the relational planning problem description to the lifted SOGBOFA graph, without considering each ground node, but we do not pursue this here. On its face the saving from this construction does not look significant because we only have one pass of messages in the computation of BP so that spending linear time in advance is already expensive. This would be correct if we only ran BP once. However, because we evaluate the same lifted SOGBOFA graph and calculate gradients over it many times in the course of optimization, the effect of a compressed computation graph is significant. The second difference is due to the inputs to the two algorithms. SOGBOFA uses algebraic expressions whereas lifted BP uses tabular CPTs. Therefore, SOGBOFA’s computation is potentially more structured than lifted BP because the input CPTs are given as algebraic expressions and because sub-expressions are also represented as nodes in the graph. In other words lifting can occur at sub-CPT level as well as at the full CPT level as in Lifted BP. Summarizing this discussion we have:

**Observation 1.** The computation graph of Lifted SOGBOFA is at least as compressed as the message structure in Lifted BP when run on the DBN generated by the planning problem, conditioned on the initial state, initial action, and rollout policy, and with no evidence.

Lifted SOGBOFA shrinks the computation graph and leads to faster estimates of action values. This can be used in two ways. Given a fixed time constraint, we can aim for the same number of updates and deeper search or keeping the same depth we can perform more updates or random restarts. Both of these have the potential to improve planning performance.

**Experimental Validation**

For the evaluation we need to discuss one additional aspect of SOGBOFA. The algorithm aims at on-line planning where one has to decide on the next action in a short amount of time, and then the process repeats. The algorithm unrolls the DBN given by a planning problem to a horizon \( h \) and then performs the optimization as described. If \( h \) is too deep the computation is slow and there is not enough time for the optimization. Therefore SOGBOFA dynamically decides on the depth in order to trade off accuracy with available computation time. Speedup in construction means that the algorithm will search to a greater depth. This potentially complicates the comparison of the original and lifted versions of the algorithm. For the experiments in this paper, we manually fix the depth for each problem to a reasonable setting, and then compare the results of the two algorithms. This evaluates the potential improvement from having more updates in the search process.

The work of (Cui and Khardon 2016) has evaluated the
algorithms on problems of varying sizes from 6 planning domains and the code and details are available at https://github.com/hcui01/SOGBOFA. We follow that evaluation in using the original code and the same problems.

We first evaluate the contribution of lifting to SOGBOFA in terms of graph size and number of updates on problem 11 in each domain. Following exploratory experiments, we fixed the depth manually to a value that was expected to yield reasonable results for SOGBOFA. The depth used, average graph size (over multiple steps), average number of updates within 10 seconds are reported in Table 1. As can be seen, on 3 of the domains, the lifted algorithm creates smaller graphs and results in more updates. On the other domains there is a small improvement but the differences are not significant. Therefore in the following experiments we focus on the 3 domains where the lifting compression looks promising.

We next turn to evaluate whether the compression translates to better planning performance. For this we follow the experimental methodology from (Cui and Khardon 2016). We run all problems with horizon 40 and each problem instance is run 20 times. The results reported represent the average cumulative reward (no discounting) and standard deviations from these 20 runs. To control the run time needed for these experiments the algorithms are given 10 seconds per time step (compared to 60 seconds in (Cui and Khardon 2016)).

The results are shown in Figure 1. For visual clarity raw scores of cumulative reward are normalized relative to two fixed algorithms. The random policy is normalized as 0 and a simple human coded policy is normalized as 1. The raw scores are linearly scaled relative to these. As can be seen in the figure lifting provides an improvement in performance across a range of problems. For skill teaching a significant improvement can be seen on problems 6-20. For tamarisk and traffic, the improvement is more modest, and there are three ranges of problem sizes. The smaller and easier problems are already solved well by SOGBOFA and we do not get a significant improvement. The largest problems are too large for both algorithms (with 10 seconds per step), i.e., they do not get enough updates and their performance degrades. In the middle (problems 11-13 on tamarisk and 11-14 on traffic) lifting provides performance improvements. This analysis is corroborated by looking at graph sizes and the number of gradient updates from the same runs. Overall, the lifted algorithm does not degrade performance on any problem and improves performance in large number of cases. The range of problems where it provides an improvement will change with the setting of time limit, where we expect the gap to move to the right and increase for large problems, when given more time.

**Algebraic Solver for Marginal MAP**

Marginal MAP is computationally hard and state of the art exact algorithms use branch and bound techniques (Lee et al. 2016). Approximation algorithms include mixed belief propagation (Liu and Ihler 2013), an extension of BP that directly addresses MMAP.

The SOGBOFA algorithm uses the DBN model to solve the planning problem. But the analogy to inference does not require the DBN structure. The main conditions that are required are that the input is a directed model, that the action nodes are roots of that model, and that there is a single value node at a leaf of the directed model. The corresponding requirements for MMAP are that max nodes are at the roots of the model and that there is a single evidence node at a leaf of this graph.

Given a MMAP problem with multiple evidence nodes, we can first disconnect the edges to children of evidence nodes by substituting the evidence node values directly. We can then connect all leaf evidence node to a new leaf node (denoted L* below) which captures the logical conjunction requiring that all those nodes have their observed values. See (Mauá 2016). This leads to the following algorithm:

### Algebraic MMAP Algorithm:

1. The domains are taken from International Probabilistic Planning Competition (IPPC) 2011 (sysadmin, traffic), IPPC 2014 (skill teaching, academic advising, tamarisk) and the RDDL distribution (elevators) (Sanner 2010)

2. In particular, we pick the average depth value in a run of SOGBOFA with dynamic depth on problem 11 in each domain.
(1) Given a MMAP problem with decision nodes at roots perform the above translation to make it suitable for inference by SOGBOFA.

(2) Generate a lifted algebraic graph from this MMAP problem, where MAP nodes are treated as action nodes and \( L^* \) is the reward node of the planning problem. This graph satisfies the requirements for the SOGBOFA optimizer.

(3) Optimization for decision nodes is done using the same procedure as in SOGBOFA, in particular, using gradient ascent on the prior marginals for the decision nodes, where gradients are computed using automatic differentiation and where gradient updates use an adaptive step size.

Preliminary experiments with this algorithm suggest that it is competitive or better than mixed belief propagation (Liu and Ihler 2013) for this class of problems. We leave a more thorough experimental evaluation to future work. An interesting question is whether these ideas can be used to solve general MMAP problems. Maú (2016) has shown that one can translate any MMAP problem into a MMAP (or maximum expected utility, MEU) problem with MAP variables at roots and a single evidence (or reward) variable. Unfortunately, while that construction is correct for exact inference, it does not preserve the advantage of the forward search in SOGBOFA. We leave an investigation into alternative constructions for future work.

**Related Work**

The connection between planning an inference is not new. In the context of influence diagrams, the connection between MEU and marginal MAP shown to be equivalent by (Mauú 2016), was pointed out by (Cooper 1988).

Starting from an AI planning formulation, Domshlak and Hoffmann (2006) have shown how the conformant planning problem can be solved using weighted model counting for CNF, and Lee, Marinescau, and Dechter (2016) have shown how the problem can be translated to MMAP and solved directly by MMAP algorithms.

Another major direction has been MDP formulations of factored state and action spaces, where symbolic versions of dynamic programming algorithms were developed (Hoey et al. 1999; Boutilier et al. 1995; Raghavan et al. 2012; 2013). It is easy to see that symbolic value iteration corresponds to alternating variable elimination where we alternate between eliminating state variables of a time slice and action variables of the same time slice. These algorithms cleverly avoid an explicit representation of the potentially exponential size policy. Both exact and approximate algorithms have been developed. The main difficulty with this approach is the space requirements of representing value functions or policies which can become prohibitive.

A different approach to MDPs has been through formulations as variational inference. Starting with (Dayan and Hinton 1997) several formulations define a reward weighted posterior distribution over trajectories, where identifying the MAP over actions gives the optimal policy. Solutions include the expectation maximization (EM) or variational EM algorithms (Toussaint and Storsky 2006; Furmston and Barber 2010; Neumann 2011; Kobé and Peters 2011) policy gradients (Kobé and Peters 2011; van de Meent et al. 2016) and BP (Liu and Ihler 2012; Cheng et al. 2013). As discussed above, one of the main differences between these approaches and ours is that they condition on the reward whereas our approach simply evaluates the marginal of the reward. While in principle these are equivalent, the computational properties of different approximation algorithms on the different graphs is not necessarily the same.

The work of Lang and Toussaint (2010) introduces an algorithm in the context of planning for robotics which, with the hindsight of this paper, looks close to AROLLOUT. Their algorithm performs action evaluation by forward simulation using the factored frontier algorithm, which is equivalent to BP. The details differ, however, because they sample concrete trajectories and do not take advantage of the symbolic aggregate simulation. Lifted SOGBOFA could potentially improve planning performance in that approach.

**Conclusions**

The paper identifies a connection between a successful heuristic for planning in large factored state and action spaces and belief propagation. The SOGBOFA heuristic performs its estimation symbolically and through that performs its search using gradients. This suggests a general scheme for approximation algorithms for MMAP. Any approximation scheme whose value or objective can be represented using an explicit computation graph, can be optimized directly with gradient search through automatic differentiation. Two immediate consequences of this connection are shown to derive new interesting algorithms. The first taking ideas from inference to planning is a lifted version of SOGBOFA which improves planning performance. The second, taking ideas from planning to inference, is a new algorithm that applies SOGBOFA inference for a subclass of MMAP problems. These connections expose the potential for improving algorithms further in both fields.

---

### Table 1: Graph size and number of update statistics for SOGBOFA and lifted SOGBOFA

<table>
<thead>
<tr>
<th>Domain</th>
<th>elevators</th>
<th>sysadmin</th>
<th>academic</th>
<th>tamarisk</th>
<th>traffic</th>
<th>skill</th>
</tr>
</thead>
<tbody>
<tr>
<td>depth</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>SOGBOFA Size</td>
<td>22103</td>
<td>4973</td>
<td>6065</td>
<td>6180.8</td>
<td>10046</td>
<td>1528.2</td>
</tr>
<tr>
<td>Lifted Size</td>
<td>22103</td>
<td>4642</td>
<td>5743.4</td>
<td>1769.2</td>
<td>5362</td>
<td>1004.6</td>
</tr>
<tr>
<td>SOGBOFA Updates</td>
<td>166.7</td>
<td>860.4</td>
<td>421</td>
<td>49.4</td>
<td>195</td>
<td>1810</td>
</tr>
<tr>
<td>Lifted Updates</td>
<td>172.3</td>
<td>937</td>
<td>807.8</td>
<td>728.2</td>
<td>411.8</td>
<td>3475.2</td>
</tr>
</tbody>
</table>

---

663
Acknowledgments
This work was partly supported by NSF under grant IIS-1616280. Some of the experiments in this paper were performed on the Tufts Linux Research Cluster supported by Tufts Technology Services.

References


Singla, P., and Domingos, P. M. 2008. Lifted first-order belief propagation. In AAAI.
