Relational Decision Theory

Abstract

In recent years, many representations have been proposed that combine graphical models with aspects of first-order logic, along with learning and inference algorithms for them. However, the problem of extending decision theory to these representations remains largely unaddressed. In this paper, we propose a framework for relational decision theory based on Markov logic, which treats weighted first-order clauses as templates for features of Markov networks. By allowing clauses to have utility weights as well as probability weights, very rich utility functions can be represented. In particular, both classical planning and Markov decision processes are special cases of this framework. Maximizing expected utility in this representation is intractable, but we develop approximate algorithms for it. Experiments on a viral marketing domain and a probabilistic combinatorial auction problem illustrate the power of our approach and the efficiency of the algorithms.

1 Introduction

Intelligent agents must be able to handle the complexity and uncertainty of the real world. First-order logic is useful for the first, and probability for the second. Combining the two has been the focus of much recent research [Getoor and Taskar, 2007]. However, there is little work to date on extending these representations to the decision-theoretic setting, which is needed to allow agents to intelligently choose actions. The one major exception is relational reinforcement learning and first-order MDPs (e.g., Džeroski and De Raedt [1998]; van Otterlo [2005]; Sanner [2008]). However, the representations and algorithms in these approaches are geared to the problem of sequential decision-making, and many decision-theoretic problems are not sequential. In particular, relational domains often lead to very large and complex decision problems for which no effective general solution is currently available (e.g., influence maximization in social networks, combinatorial auctions with uncertain supplies, etc.).

The goal of this paper is thus to provide a general decision-theoretic extension of first-order probabilistic representations and their inference algorithms. Our starting point is Markov logic, one of the most powerful representations available [Richardson and Domingos, 2006]. We extend it to represent decision-theoretic problems, and develop an efficient algorithm for maximizing expected utility in this formulation, based on lifted belief propagation [Singla and Domingos, 2008]. We provide theoretical guarantees for our algorithm. Experiments in viral marketing and combinatorial auction domains show that (a) these problems can be elegantly formulated in our language, and (b) our inference algorithm is much more efficient than a direct application of lifted BP.

2 Background

2.1 Markov Networks and Decision Theory

Graphical models compactly represent the joint distribution of a set of variables $\mathbf{X} = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$ as a product of factors [Pearl, 1988]: $P(\mathbf{X}{=}\mathbf{x}) = \frac{1}{Z}\prod_k f_k(\mathbf{x}_k)$, where each factor f_k is a non-negative function of a subset of the variables $\mathbf{x}_k$, and Z is a normalization constant. Under appropriate restrictions, the model is a *Bayesian network* and $Z = 1$. A *Markov network* or *Markov random field* can have arbitrary factors. Graphical models can also be represented in *log-linear form*: $P(\mathbf{X}{=}\mathbf{x}) = \frac{1}{Z}\exp\left(\sum_i w_i g_i(\mathbf{x})\right)$, where the *features* $g_i(\mathbf{x})$ are arbitrary functions of the state.

A key inference task in graphical models is computing the marginal probabilities of some variables (the query) given the values of some others (the evidence). This problem is #P-complete, but can be solved approximately using *loopy belief propagation* (BP) [Weiss, 2000]. In the simplest form, the Markov network is first converted to an equivalent pairwise network. Belief propagation then works by repeatedly passing messages between nodes in this network. The message from node t to node s is $m_{ts}(x_s) = \sum_{\sim\{x_s\}}(\phi_{ts}(x_t, x_s)\phi_t(x_t)\prod_{u\in nb(t)\setminus\{s\}} m_{ut}(x_t))$, where $nb(t)$ is the set of neighbors of t, and the sum is over all of these except s. The (unnormalized) belief at node t is given by $M_t = \phi_t(x_t)\prod_{u\in nb(t)} m_{ut}(x_t)$. Many message-passing schedules are possible; the most widely used one (and generally the most accurate) is *flooding*, where all nodes send messages at each step. In general, belief propagation is not guaranteed to converge, and it may converge to an incorrect result, but in practice it often approximates the true probabilities well.

An *influence diagram* or *decision network* is a graphical representation of a decision problem [Howard and Matheson, 2005]. It consists of a Bayesian network augmented with two types of nodes: *decision* or *action* nodes and *utility* nodes. The action nodes represent the agent's choices; factors involving these nodes and *state* nodes in the Bayesian network represent the (probabilistic) effect of the actions on the world. *Utility* nodes represent the agent's utility function, and are connected to the state nodes that directly influence utility. We can also define a *Markov decision network* as a decision network with a Markov network instead of a Bayesian network.

The fundamental inference problem in decision networks is finding the assignment of values to the action nodes that maximizes the agent's expected utility, possibly conditioned on some evidence. If $\mathbf{a}$ is a choice of actions, $\mathbf{e}$ is the evidence, $\mathbf{x}$ is a state, and $U(\mathbf{a}|\mathbf{e})$ is the utility of $\mathbf{a}$ given $\mathbf{e}$, then the *MEU problem* is to compute $\operatorname{argmax}_{\mathbf{a}} E[U(\mathbf{a}|\mathbf{e})] = \operatorname{argmax}_{\mathbf{a}} \sum_{\mathcal{X}} P(\mathbf{X} = \mathbf{x}|\mathbf{a}, \mathbf{e})U(\mathbf{x})$.

2.2 Markov Logic

Markov logic is a probabilistic extension of first-order logic. Formulas in first-order logic are constructed from logical connectives, predicates, constants, variables and functions. A *grounding* of a predicate (or *ground atom*) is a replacement of all its arguments by constants (or, more generally, ground terms). Similarly, a grounding of a formula is a replacement of all its variables by constants. A *possible world* is an assignment of truth values to all possible groundings of predicates.

A *Markov logic network (MLN)* is a set of weighted first-order formulas. Together with a set of constants, it defines a Markov network with one node per ground atom and one feature per ground formula. The weight of a feature is the weight of the first-order formula that originated it. The probability distribution over possible worlds $\mathbf{x}$ specified by the MLN and constants is thus $P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp(\sum_i w_i n_i(\mathbf{x}))$, where w_i is the weight of the ith formula and $n_i(\mathbf{x})$ its number of true groundings in $\mathbf{x}$.

Inference in an MLN can be carried out by creating the corresponding ground Markov network and applying standard belief propagation to it. A more efficient alternative is *lifted belief propagation*, which avoids grounding the network as much as possible [Singla and Domingos, 2008]. Lifted BP constructs a *lifted network* composed of *supernodes* and *superfeatures*, and applies BP to it. A supernode is a set of ground atoms that all send and receive exactly the same messages throughout BP, and a superfeature is defined analogously.

3 Markov Logic Decision Networks

Decision theory can be incorporated into Markov logic simply by allowing formulas to have utilities as well as weights. This puts the expressiveness of first-order logic at our disposal for defining utility functions, at the cost of very little additional complexity in the language. Let an *action predicate* be a predicate whose groundings correspond to possible actions (choices, decisions) by the agent, and a *state predicate* be any predicate in a standard MLN. Formally:

Definition A *Markov logic decision network (MLDN)* L is a set of triples (F_i, w_i, u_i), where F_i is a formula in first-order logic and w_i and u_i are real numbers. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov decision network $M_{L,C}$ as follows:

1. $M_{L,C}$ contains one binary node for each possible grounding of each state and action predicate appearing in L. The value of the node is 1 if the ground atom is true, and 0 otherwise.
2. $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L for which $w_i \neq 0$. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.
3. $M_{L,C}$ contains one utility node for each possible grounding of each formula F_i in L for which $u_i \neq 0$. The value of the node is the utility u_i associated with F_i in L if F_i is true, and 0 otherwise.

We refer to formulas with non-zero weight as *probability formulas*, and formulas with non-zero utility as *utility formulas*. Groundings of action predicates are *action atoms*, and groundings of state predicates are *state atoms*. An assignment of truth values to all action atoms is an *action choice*. An assignment of truth values to all state atoms is a *state of the world* or *possible world*. The utility of a possible world $\mathbf{x}$ is $U(\mathbf{X} = \mathbf{x}) = \sum_i u_i n_i(\mathbf{x})$, where n_i is the number of true groundings of F_i in $\mathbf{x}$. The expected utility of action choice $\mathbf{a}$ given evidence $\mathbf{e}$ is

$$E[U(\mathbf{a}|\mathbf{e})] = \sum_{\mathbf{x}} P(\mathbf{x}|\mathbf{a}, \mathbf{e}) \sum_i u_i n_i(\mathbf{x}) = \sum_i u_i E[n_i].$$

The MEU problem in MLDNs is finding the action choice that maximizes expected utility, and is obviously intractable. The next section deals with efficient approximate algorithms for solving it.

A wide range of decision problems can be elegantly formulated as MLDNs, including both classical planning and Markov decision processes (MDPs).

To represent an MDP as an MLDN, we can define a constant for each state, action and time step, and the predicates $\texttt{State}(\texttt{s!}, \texttt{t})$ and $\texttt{Action}(\texttt{a!}, \texttt{t})$, with the obvious meaning. (The ! notation indicates that, for each $\texttt{t}$, exactly one grounding of $\texttt{State}(\texttt{s}, \texttt{t})$ is true, and similarly for actions [Kok *et al.*, 2008].) The transition function is then represented by the formula $\texttt{State}(+\texttt{s}, \texttt{t}) \wedge \texttt{Action}(+\texttt{a}, \texttt{t}) \Rightarrow \texttt{State}(+\texttt{s}', \texttt{t} + 1)$, with a separate weight for each $(\texttt{s}, \texttt{a}, \texttt{s}')$ triple. (We follow the convention of using a formula with $+$ signs before certain variables to represent a set of identical formulas with separate weights, one for each combination of groundings of the variables with $+$ signs [Kok *et al.*, 2008].) The reward function is defined by the unit clause $\texttt{State}(*\texttt{s}, \texttt{t})$, with a utility for each state (using $*$ to represent per-grounding utilities). Policies can be represented by formulas of the form $\texttt{State}(+\texttt{s}, \texttt{t}) \Rightarrow \texttt{Action}(+\texttt{a}, \texttt{t})$. Infinite-horizon MDPs can be represented using infinite MLNs [Singla and Domingos, 2007]. Partially-observable MDPs are represented by adding the observation model: $\texttt{State}(+\texttt{s}, \texttt{t}) \Rightarrow \texttt{Observation}(+\texttt{o}, \texttt{t})$.

Since classical planning languages are variants of first-order logic, translating problems formulated in these languages into MLDNs is straightforward. For simplicity, suppose the problem has been expressed in satisfiability form [Kautz and Selman, 1992]. It suffices then to translate the CNF into a (deterministic) MLN by assigning infinite weight to all clauses, and to assign a positive utility to the formula defining the goal states. MLDNs now offer a path to extend classical planning with uncertain actions, complex utilities, etc., by assigning finite weights and utilities to formulas. (For example, an action with uncertain effects can be represented by assigning a finite weight to the axiom that defines them.) This can be used to represent first-order MDPs in a manner analogous to Boutilier *et al.* [2001].

4 Maximizing Expected Utility

In principle, the MEU action choice can be found by searching exhaustively over all action choices, computing the expected utility of each using any inference method. However, this will be infeasible in all but the smallest domains. An obvious alternative, particularly in very large domains, is greedy search: starting from a random action choice, consider flipping each action atom in turn, do so if it increases expected utility, and stop when a complete cycle produces no improvements. This is guaranteed to converge to a local optimum of the expected utility, and is the method we use in our experiments. There is a subtlety, however: because MLDNs can include arbitrary hard constraints on the action predicates, flipping a single action atom may not lead to a feasible solution. In general, a minimum set of flips that leads to a feasible solution must be found, and this can be done using a satisfiability solver. In many cases, however, much simpler methods will suffice. For example, in the common case where exactly one of n actions must be performed, we can simply try each one and choose the best.

The expected utility of action choices can be computed by grounding the MLDN and running belief propagation. A potentially much more efficient alternative is to use lifted BP, as described in Section 2.2. This can be done by constructing the lifted network as before, treating action predicates as non-evidence predicates, and then searching over *superaction* choices, a superaction being a set of action atoms that were grouped together as a supernode by the lifted network construction algorithm. These atoms are guaranteed to all be set to the same truth value in any globally or locally optimal solution, since by construction they all have exactly the same effect on the expected utility.

However, even with lifted BP such an approach will generally be extremely inefficient. This is because it will need to perform a complete run of BP for each action choice tried, of which there could easily be millions or more. However, most of this computation will typically be redundant, because in a large network flipping a single action atom is unlikely to significantly change the probabilities of most state atoms. We have developed an algorithm that takes advantage of this, which we call *expanding frontier belief propagation (EFBP)*. EFBP starts by computing the utility of the initial action choice using standard BP (e.g., by flooding). Then, at each greedy search step, it maintains a set T of nodes affected by the changed action atom(s), initialized to contain only those changed atoms. In each iteration of BP, only the nodes in T send messages. Neighbors of nodes in T are added to T if the messages they receive differ by more than a threshold γ from the final messages they received when they last participated in BP. (The neighbors of a node are the nodes that appear in some factor with it.) In the worst case, the whole network may be added to T, and we revert to BP. In many domains, however, the effect of a change usually dies down quickly, only influencing a small portion of the network. In such situations, EFBP can converge much faster than BP.

EFBP is based on a similar principle to residual belief propagation (RBP) [Elidan *et al.*, 2006]: focus effort on the portions of the graph that are furthest from convergence. However, while RBP is used to schedule message updates in a single run of belief propagation, the purpose of EFBP is to minimize redundant computation when repeatedly running BP on the same network, with varying settings of some of the variables. In this sense, EFBP is an approximate algorithm for adaptive inference [Acar *et al.*, 2008]. One could run EFBP using RBP to schedule messages, to speed up convergence.

If the potentials are bounded, EFBP's estimate of the expected utility is guaranteed to be close to BP's. We first show that EFBP's marginal probability estimates provide bounds on BP's. We can do this by viewing the difference between the beliefs generated by EFBP and those generated by BP as multiplicative error in the messages passed by EFBP:

$$\hat{m}^i_{ts}(x_s) = m^i_{ts}(x_s) e^i_{ts}(x_s),$$

where $\hat{m}^i_{ts}(x_s)$ is the message sent by EFBP in iteration i (i.e., nodes not in T resend the same messages as in $i-1$); $m^i_{ts}(x_s)$ is the message sent if all nodes recalculate their messages in iteration i, as in BP; and $e^i_{ts}(x_s)$ is the multiplicative error introduced in iteration i of EFBP.

The *dynamic range* of a function is defined as follows [Ihler *et al.*, 2005]:

$$d(f) = \sup_{x,y} \sqrt{f(x)/f(y)}$$

Since EFBP recalculates its messages when $|\hat{m}^i_{ts} - m^i_{ts}| > \gamma$, $m^i_{ts}(x_s) - \gamma \le m^i_{ts}(x_s) e^i_{ts}(x_s) \le m^i_{ts}(x_s) + \gamma$, and therefore $1 - \gamma/m^i_{ts}(x_s) \le e^i_{ts}(x_s) \le 1 + \gamma/m^i_{ts}(x_s)$. This allows us to bound the dynamic range of e^i_{ts}:

$$d(e^i_{ts}) \le \sup_{x,y} \sqrt{\frac{1 + \gamma/\left(m^i_{ts}(x_s)\right)}{1 - \gamma/\left(m^i_{ts}(y_s)\right)}} = \delta(e^i_{ts})$$

Theorem 15 of Ihler *et al.* [2005] implies that, for any fixed point beliefs $\{M_t\}$ found by belief propagation, after $n \ge 1$ iterations of EFBP resulting in beliefs $\{\hat{M}^n_t\}$ we have

$$\log d(M_t/\hat{M}^n_t) \le \sum_{u \in nb(t)} \log \nu^n_{ut} = \log \zeta^n_t, \qquad (1)$$

where $\log \nu^1_{ts} = \delta(e^1_{ts}) d(\phi_{ts})^2$, and ν^i_{ut} is given by:

$$\log \nu^{i+1}_{ts} = \log \frac{d(\phi_{ts})^2 \varepsilon^i_{ts} + 1}{d(\phi_{ts})^2 + \varepsilon^i_{ts}} + \log \delta(e^i_{ts})$$

$$\log \varepsilon^i_{ts} = \sum_{u \in nb(t) \setminus s} \log \nu^i_{ut}$$

Intuitively, ν^i_{ut} can be thought of as a measure of the accumulated error in the incoming message from u to t in iteration i. It can be computed iteratively using a message-passing algorithm similar to BP.

Lemma 4.1. *For state atom x_t, the probability estimated by BP at convergence (p_t) can be bounded as follows in terms of the probability estimated by EFBP ($\hat{p}_t$) after n iterations:*

$$p_t \geq \frac{1}{(\zeta^n_t)^2[(1/\hat{p}_t)-1]+1} = lb(p_t)$$
$$p_t \leq \frac{1}{(1/\zeta^n_t)^2[(1/\hat{p}_t)-1]+1} = ub(p_t)$$

Proof. From Equation 1, $d(M_t/\hat{M}^n_t) \leq \zeta^n_t$, and therefore $\frac{M_t(1)/\hat{M}^n_t(1)}{M_t(0)/\hat{M}^n_t(0)} \leq (\zeta^n_t)^2$ and $(1-\hat{p}_t)/\hat{p}_t \leq (\zeta^n_t)^2(1-p_t)/p_t$, where p_t and $\hat{p}_t$ are obtained by normalizing M_t and $\hat{M}_t$. The upper bound follows, and the lower bound can be obtained similarly. □

Theorem 4.2. *The expected utility $E_{bp}[U]$ of an action choice estimated by BP can be bounded as follows:*

$$E_{bp}[U] \geq \sum_{t\in U^+} lb(p_t)u_t + \sum_{t\in U^-} ub(p_t)u_t$$
$$E_{bp}[U] \leq \sum_{t\in U^+} ub(p_t)u_t + \sum_{t\in U^-} lb(p_t)u_t$$

where U^+ and U^- are the utility nodes with positive and negative utility, respectively, u_t is the utility of the utility node t, and p_t is the marginal probability of the corresponding utility formula.

Proof. Add to the network a new state node for each utility formula F, with truth value constrained by a hard formula to be equal to the truth value of F (i.e., a new factor is added, involving the new node and the ones that appear in F). The theorem then follows immediately from Lemma 4.1 and the definition of expected utility. □

Based on this theorem, we can design a variant of EFBP that is guaranteed to produce the same action choice as BP when used with greedy search. Let $\mathbf{A}_i$ denote the set of action choices considered in the ith step of greedy search. If EFBP picks action choice $\mathbf{a} \in \mathbf{A}_i$, BP is guaranteed to make the same choice if the following condition holds:

$$lb(E[U(\mathbf{a})]) > \max_{\mathbf{a}'\in\mathbf{A}_i} ub(E[U(\mathbf{a}')]) \quad (2)$$

When the condition does not hold, we revert the messages of all nodes to their values after the initial BP run, insert all changed action atoms into T, and rerun EFBP. If condition 2 still does not hold, we halve γ and repeat. (If γ becomes smaller than the BP convergence threshold, we run standard BP.) We call this algorithm EFBP$*$.

Note that using the above bound for search requires a slight modification to the calculation of ν^1_{ts}, since EFBP and BP result in different message initializations for the first BP iteration at every search step. If BP initializes all messages to 1 before every search step, the error function $e^1_{ts}(x_s) = \hat{m}^1_{ts}(x_s)$. If BP initializes its messages with their values from end of the previous search iteration, then ν^1_{ts} can also be similarly initialized. The calculations of ν^i_{ts} for $i \neq 1$ are not altered.

Let Greedy(I) denote greedy search using inference algorithm I to compute expected utilities.

Theorem 4.3. *Greedy(EFBP$*$) outputs the same action choice as Greedy(BP).*

Proof. Condition 2 guarantees that EFBP makes the same action choice as BP. Since EFBP$*$ guarantees that the condition holds in every iteration of the final run of EFBP, it guarantees that the action choice produced is the same. □

In practice, EFBP$*$ is unlikely to provide large speedups over BP, since the bound in Lemma 4.1 must be repeatedly recalculated even for nodes outside T. However, empirically, running EFBP with a fixed threshold not much higher than BP's convergence threshold seems to yield an action choice of very similar utility to BP's in a fraction of the time.

5 Experiments

Our experiments had two goals: to find out how well MLDNs can model a variety of decision problems in relational, uncertain domains, and to evaluate the performance of EFBP. We used two domains: viral marketing and combinatorial auctions. We implemented MLDNs and EFBP as extensions of the Alchemy system [Kok *et al.*, 2008]. The experiments were run on a cluster of 8-processor machines with 2GB of RAM per processor, running at 2.33 GHz. We used a convergence threshold of 10^{-4} for flooding, including the initial BP run for EFBP, and a threshold of $\gamma = 10^{-3}$ for the remaining iterations of EFBP. We evaluated the utility of the actions chosen by EFBP using a second run of full BP, with a threshold of 10^{-4}. As is usually done, both algorithms were run for a few (10) iterations after the convergence criterion was met.

5.1 Viral Marketing

Viral marketing is based on the premise that members of a social network influence each other's purchasing decisions. The goal is then to select the best set of people to market to, such that the overall profit is maximized by propagation of influence through the network. Originally formalized by Domingos and Richardson [2001], this problem has since received much attention, including both empirical and theoretical results.

A standard dataset in this area is the Epinions web of trust [Richardson and Domingos, 2002]. Epinions.com is a knowledge-sharing Web site that allows users to post and read reviews of products. The "web of trust" is formed by allowing users to maintain a list of peers whose opinions they trust. We used this network, containing 75,888 users and over 500,000 directed edges, in our experiments. With over 75,000 action nodes, this is a very large decision problem, and no general-purpose MEU algorithms have previously been applied to it (only domain-specific implementations).

We defined an MLDN very similar to Domingos and Richardson's [2001] model using the state predicates `Buys(x)` and `Trusts(x1, x2)`, and the action predicate `MarketTo(x)`. The utility function is represented by the unit

clauses $\texttt{Buys}(\texttt{x})$ (with positive utility, representing profits from sales) and $\texttt{MarketTo}(\texttt{x})$ (with negative utility, representing the cost of marketing). The topology of the social network is specified by an evidence database of $\texttt{Trusts}(\texttt{x}_1, \texttt{x}_2)$ atoms. Information about who has already bought the product can also be incorporated, in the form of $\texttt{Buys}(\texttt{x})$ evidence atoms.

The core of the model consists of two formulas:

$$\texttt{Buys}(+\texttt{x}_1) \wedge \texttt{Trusts}(+\texttt{x}_2, \texttt{x}_1) \Rightarrow \texttt{Buys}(\texttt{x}_2) \quad (1)$$

$$\texttt{MarketTo}(+\texttt{x}) \Rightarrow \texttt{Buys}(\texttt{x}) \quad (2)$$

In addition, the model includes the unit clause $\texttt{Buys}(\texttt{x})$ with a negative weight, representing the fact that most users do not buy most products. The weight of Formula 1 for user pair $(\texttt{x}_2, \texttt{x}_1)$ represents how strongly $\texttt{x}_1$ influences $\texttt{x}_1$, and the weight of Formula 2 represents how strongly users are influenced by marketing. These parameters can be estimated from data, as in Domingos and Richardson [2001]. For our experiments, however, we varied the weights of the formulas, to see how this affected the behavior of the algorithms.

We compared BP and EFBP, with and without lifting. We set the cost of marketing to each user to -1, and the profit from each purchase to 20. The weight of the $\texttt{Buys}(\texttt{x})$ formula was -2. The initial action choice was to market to no one. All results are averages of six runs. Fixing the weight of formula 2 at 0.8 for all users, inference times varied as shown in Figure 1(a) as we changed the weight of Formula 1. Running utility maximization with BP until convergence was not feasible; instead, we extrapolated from the number of actions considered during the first 80 hours, assuming that search with BP would consider the same number of actions as search with EFBP. Figure 1(b) plots the result of a similar experiment, with influence weight fixed at 0.6 and varying the weight of Formula 2. In both cases, EFBP was consistently orders of magnitude faster than BP. Lifted EFBP further reduced running times (notice the difference is obscured by the log scale).

We can also compare the utility obtained by BP and EFBP given a fixed running time of 80 hours. EFBP consistently achieves about 40% higher utility than BP when varying the influence weight, with higher advantage for lower weights. For high marketing weights, EFBP achieves about 30% higher utility than BP; this advantage decreases gradually to zero as the weight is reduced (reflecting the fact that fewer and fewer customers buy).

Richardson and Domingos [2002] tested various specially designed algorithms on the same network. They estimated that inference using the model and algorithm of Domingos and Richardson [2001], which are the most directly comparable to ours, would have taken hundreds of hours to converge (100 per pass). (This was reduced to 10-15 minutes using additional problem-specific approximations, and a much simpler linear model that did not require search was even faster.) Although these convergence times cannot be directly compared with our own (due to the different hardware, parameters, etc., used), it is noteworthy that EFBP converges faster than the most general of their methods.

Unlike previous hand-coded models, our MLDN can be easily extended to incorporate customer and product attributes, purchase history information, multiple types of relationships, products, actors in the network, marketing actions, etc. Doing so is a direction for future work.

5.2 Probabilistic Combinatorial Auctions

Combinatorial auctions can be used to solve a wide variety of resource allocation problems [Cramton *et al.*, 2006]. An auction consists of a set of bids, each of which is a set of requested products and an offered reward. The seller determines which bids to assign each product to. When all requested products are assigned to a bid, the bid is said to be satisfied, and the seller collects the reward. The seller's goal is to choose an assignment of products to bids that maximizes the sum of the rewards of satisfied bids. While there has been much research on combinatorial auctions, it generally assumes that the world is deterministic (with a few exceptions, e.g., Golovin [2007]). In practice, however, both the supply and demand for products are subject to many sources of uncertainty. (For example, supply of one product may make supply of another less likely because they compete for resources.)

We model uncertainty in combinatorial auctions by allowing product assignments to fail, resulting in the loss of reward from the corresponding bids. The following MLDN describes our formulation of the problem.

$$\forall\, \texttt{product} : \texttt{InBid}(\texttt{product}, *\texttt{bid}) \Rightarrow \texttt{Assign}(\texttt{product}, \texttt{bid}) \wedge \texttt{Succeed}(\texttt{product}) \quad (1)$$

$$\texttt{Competes}(+\texttt{product1}, +\texttt{product2}) \Rightarrow (\neg\texttt{Succeed}(\texttt{product1}) \vee \neg\texttt{Succeed}(\texttt{product2})) \quad (2)$$

$\texttt{InBid}(\texttt{product}, \texttt{bid})$ and $\texttt{Competes}(\texttt{product}, \texttt{product})$ are evidence predicates. $\texttt{Assign}(\texttt{product}, \texttt{bid})$ is an action predicate. Formula 1 is a utility formula; each grounding represents a single bid. Formula 2 represents the supply uncertainty. It models competition between two products; the success of one increases the failure probability of the other. This formula creates a network of dependencies between product failures.

We generated bids according to the decay distribution described by Sandholm [2002], with $\alpha = 0.75$, and randomly generated 1000 true $\texttt{Competes}()$ atoms from a uniform distribution over product pairs. Figure 1(c) plots the inference time for a 1000-product, 1000-bid auction, varying the weight of Formula 1. The results are averages of ten runs. As in the viral marketing experiment, EFBP converges much faster than BP. Since in this case both algorithms can be run until convergence, the expected utilities of the chosen product assignments are extremely similar.

Our MLDN can be easily extended to incorporate more interesting bidding languages and more complex probabilistic dependencies. For instance, each bid could be an arbitrary logical formula, rather than a conjunction. Or it could be a probability distribution depending on several products, representing a form of demand uncertainty. MLDNs could also be used to introduce other kinds of supply and demand uncertainty to combinatorial auction problems.

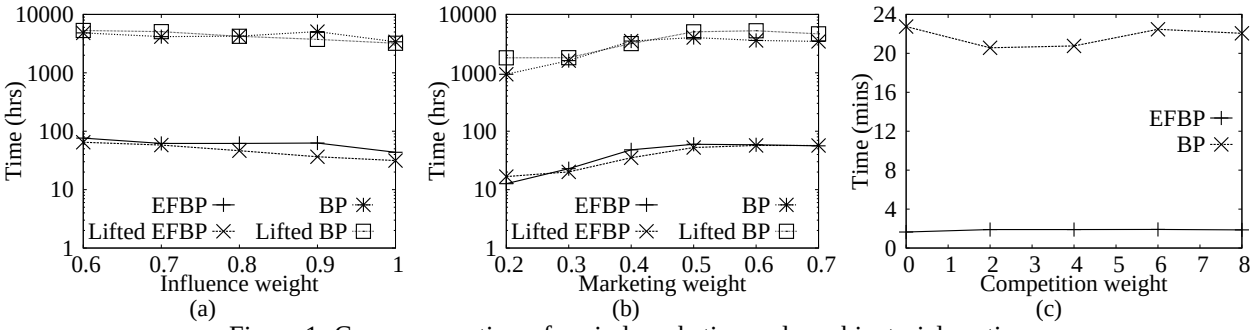

Figure 1: Convergence times for viral marketing and combinatorial auctions.

6 Conclusion

In this paper, we extended decision theory to relational domains by adding utility weights to Markov logic formulas. We call the resulting representation *Markov logic decision networks (MLDNs)*. We then developed EFBP, an efficient algorithm for inference in MLDNs based on lifted belief propagation, and provided theoretical guarantees for it. Applications to viral marketing and combinatorial auctions provided evidence of the flexibility of MLDNs and the efficiency of EFBP.

Directions for future work include: applying MLDNs to other domains, including planning ones; combining EFBP with algorithms for inference in first-order MDPs, and with domain-specific algorithms (e.g., local search algorithms for solving combinatorial auctions); extending BP and EFBP to better handle hard constraints; developing further algorithms for MEU inference in MLDNs; using MLDNs for utility-guided learning, including relational reinforcement learning; etc.

References

[Acar *et al.*, 2008] U. A. Acar, A. T. Ihler, R. R. Mettu, and Ö. Ŝumer. Adaptive inference on general graphical models. In *UAI*, 2008.

[Boutilier *et al.*, 2001] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. In *IJCAI*, 2001.

[Cramton *et al.*, 2006] P. Cramton, Y. Shoham, and R. Steinberg. *Combinatorial Auctions*. MIT Press, Cambridge, MA, 2006.

[Domingos and Richardson, 2001] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001.

[Džeroski and De Raedt, 1998] S. Džeroski and L. De Raedt. Relational reinforcement learning. In *ICML*, 1998.

[Elidan *et al.*, 2006] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *UAI*, 2006.

[Getoor and Taskar, 2007] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA, 2007.

[Golovin, 2007] D. Golovin. Stochastic packing-market planning. In *EC*, 2007.

[Howard and Matheson, 2005] R. A. Howard and J. E. Matheson. Influence diagrams. *Decision Analysis*, 2(3):127–143, 2005.

[Ihler *et al.*, 2005] Alexander T. Ihler, John W. Fisher, and Alan S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, 2005.

[Kautz and Selman, 1992] H. Kautz and B. Selman. Planning as satisfiability. In *ECAI*, 1992.

[Kok *et al.*, 2008] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, J. Wang, and P. Domingos. The Alchemy system for statistical relational AI. Technical report, University of Washington, 2008. http://alchemy.cs.washington.edu.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[Richardson and Domingos, 2002] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, 2002.

[Richardson and Domingos, 2006] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[Sandholm, 2002] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[Sanner, 2008] S. Sanner. *First-Order Decision-Theoretic Planning in Structured Relational Environments*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 2008.

[Singla and Domingos, 2007] P. Singla and P. Domingos. Markov logic in infinite domains. In *UAI*, 2007.

[Singla and Domingos, 2008] P. Singla and P. Domingos. Lifted first-order belief propagation. In *AAAI*, 2008.

[van Otterlo, 2005] M. van Otterlo. A survey of reinforcment learning in relational domains. Technical report, University of Twente, 2005.

[Weiss, 2000] G. M. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.