# Dynamic Probabilistic Relational Models

**Sumit Sanghai** **Pedro Domingos** **Daniel Weld**
Department of Computer Science and Engineeering
University of Washington
Seattle, WA 98195-2350
{sanghai,pedrod,weld}@cs.washington.edu

## Abstract

Intelligent agents must function in an uncertain world, containing multiple objects and relations that change over time. Unfortunately, no representation is currently available that can handle all these issues, while allowing for principled and efficient inference. This paper addresses this need by introducing dynamic probabilistic relational models (DPRMs). DPRMs are an extension of dynamic Bayesian networks (DBNs) where each time slice (and its dependences on previous slices) is represented by a probabilistic relational model (PRM). Particle filtering, the standard method for inference in DBNs, has severe limitations when applied to DPRMs, but we are able to greatly improve its performance through a form of relational Rao-Blackwellisation. Further gains in efficiency are obtained through the use of abstraction trees, a novel data structure. We successfully apply DPRMs to execution monitoring and fault diagnosis of an assembly plan, in which a complex product is gradually constructed from subparts.

## 1 Introduction

Sequential phenomena abound in the world, and uncertainty is a common feature of them. Currently the most powerful representation available for such phenomena is dynamic Bayesian networks, or DBNs [Dean and Kanazawa, 1989]. DBNs represent the state of the world as a set of variables, and model the probabilistic dependencies of the variables within and between time steps. While a major advance over previous approaches, DBNs are still unable to compactly represent many real-world domains. In particular, domains can contain multiple objects and classes of objects, as well as multiple kinds of relations among them; and objects and relations can appear and disappear over time. For example, manufacturing plants assemble complex artifacts (e.g., cars, computers, aircraft) from large numbers of component parts, using multiple kinds of machines and operations. Capturing such a domain in a DBN would require exhaustively representing all possible objects and relations among them. This raises two problems. The first one is that the computational cost of using such a DBN would likely be prohibitive. The second is that reducing the rich structure of the domain to a very large, "flat" DBN would render it essentially incomprehensible to human beings. This paper addresses these two problems by introducing an extension of DBNs that exposes the domain's relational structure, and by developing methods for efficient inference in this representation.

Formalisms that can represent objects and relations, as opposed to just variables, have a long history in AI. Recently, significant progress has been made in combining them with a principled treatment of uncertainty. In particular, probabilistic relational models or PRMs [Friedman *et al.*, 1999] are an extension of Bayesian networks that allows reasoning with classes, objects and relations. The representation we introduce in this paper extends PRMs to sequential problems in the same way that DBNs extend Bayesian networks. We thus call it *dynamic probabilistic relational models*, or DPRMs. We develop an efficient inference procedure for DPRMs by adapting Rao-Blackwellised particle filtering, a state-of-the-art inference method for DBNs [Murphy and Russell, 2001]. We introduce *abstraction trees* as a data structure to reduce the computational cost of inference in DPRMs.

Early fault detection in complex manufacturing processes can greatly reduce their cost. In this paper we apply DPRMs to monitoring the execution of assembly plans, and show that our inference methods scale to problems with over a thousand objects and thousands of steps. Other domains where we envisage DPRMs being useful include robot control, vision in motion, language processing, computational modeling of markets, battlefield management, cell biology, ecosystem modeling, and the Web.

The rest of the paper is structured as follows. The next two sections briefly review DBNs and PRMs. We then introduce DPRMs and methods for inference in them. The following section reports on our experimental study in assembly plan monitoring. The paper concludes with a discussion of related and future work.

## 2 Dynamic Bayesian Networks

A *Bayesian network* encodes the joint probability distribution of a set of variables, $\{Z_1, \ldots, Z_d\}$, as a directed acyclic graph and a set of conditional probability models. Each node corresponds to a variable, and the model associated with it allows us to compute the probability of a state of the variable given the state of its parents. The set of parents of $Z_i$, denoted $Pa(Z_i)$, is the set of nodes with an arc to $Z_i$ in the graph. The structure of the network encodes the as-

sertion that each node is conditionally independent of its non-descendants given its parents. The probability of an arbitrary event $Z = (Z_1, \ldots, Z_d)$ can then be computed as $P(Z) = \prod_{i=1}^{d} P(Z_i|Pa(Z_i))$.

*Dynamic Bayesian Networks (DBNs)* are an extension of Bayesian networks for modeling dynamic systems. In a DBN, the state at time $t$ is represented by a set of random variables $Z_t = \{Z_{1,t}, \ldots, Z_{d,t}\}$. The state at time $t$ is dependent on the states at previous time steps. Typically, we assume that each state only depends on the immediately preceding state (i.e., the system is first-order Markovian), and thus we need to represent the transition distribution $P(Z_{t+1}|Z_t)$. This can be done using a two-time-slice Bayesian network fragment (2TBN) $B_{t+1}$, which contains variables from $Z_{t+1}$ whose parents are variables from $Z_t$ and/or $Z_{t+1}$, and variables from $Z_t$ without any parents. Typically, we also assume that the process is stationary, i.e., the transition models for all time slices are identical: $B_1 = B_2 = \ldots = B_t = B_{\rightarrow}$. Thus a DBN is defined to be a pair of Bayesian networks $(B_0, B_{\rightarrow})$, where $B_0$ represents the initial distribution $P(Z_0)$, and $B_{\rightarrow}$ is a two-time-slice Bayesian network, which as discussed above defines the transition distribution $P(Z_{t+1}|Z_t)$.

The set $Z_t$ is commonly divided into two sets: the unobserved state variables $X_t$ and the observed variables $Y_t$. The observed variables $Y_t$ are assumed to depend only on the current state variables $X_t$. The joint distribution represented by a DBN can then be obtained by unrolling the 2TBN:

$$\begin{aligned} & P(X_0, X_1, ..., X_T, Y_0, Y_1, ..., Y_T) \\ = \quad & P(X_0)P(Y_0|X_0) \prod_{t=1}^{T} P(X_t|X_{t-1})P(Y_t|X_t) \end{aligned}$$

Various types of inference in DBNs are possible. One of the most useful is state monitoring (also known as filtering or tracking), where the goal is to estimate the current state of the world given the observations made up to the present, i.e., to compute the distribution $P(X_T|Y_0, Y_1, ..., Y_T)$. Proper state monitoring is a necessary precondition for rational decision-making in dynamic domains. Inference in DBNs is NP-complete, and thus we must resort to approximate methods, of which the most widely used one is *particle filtering* [Doucet *et al.*, 2001]. Particle filtering is a stochastic algorithm which maintains a set of particles (samples) $x_t^1, x_t^2, \ldots, x_t^N$ to approximately represent the distribution of possible states at time $t$ given the observations. Each particle $x_t^i$ contains a complete instance of the current state, i.e., a sampled value for each state variable. The current distribution is then approximated by

$$P(X_T = x|Y_0, Y_1, ..., Y_T) = \frac{1}{N} \sum_{i=1}^{N} \delta(x_T^i = x)$$

where $\delta(x_T^i = x)$ is 1 if the state represented by $x_T^i$ is same as $x$, and 0 otherwise. The particle filter starts by generating $N$ particles according to the initial distribution $P(X_0)$. Then, at each step, it first generates the next state $x_{t+1}^i$ for each particle $i$ by sampling from $P(X_{t+1}^i|X_t^i)$. It then weights these samples according to the likelihood they assign to the observations, $P(Y_{t+1}|X_{t+1}^i)$, and resamples $N$ particles from this weighted distribution. The particles will thus tend to stay clustered in the more probable regions of the state space, according to the observations.

Although particle filtering has scored impressive successes in many practical applications, it also has some significant limitations. One that is of particular concern to us here is that it tends to perform poorly in high-dimensional state spaces. This is because the number of particles required to maintain a good approximation to the state distribution grows very rapidly with the dimensionality. This problem can be greatly attenuated by analytically marginalizing out some of the variables, a technique known as *Rao-Blackwellisation* [Murphy and Russell, 2001]. Suppose the state space $X_t$ can be divided into two subspaces $U_t$ and $V_t$ such that $P(V_t|U_t, Y_1, \ldots, Y_t)$ can be computed analytically and efficiently. Then we only need to sample from the smaller space $U_t$, requiring far fewer particles to obtain the same degree of approximation. Each particle is now composed of a sample from $P(U_t|Y_1, \ldots, Y_t)$ plus a parametric representation of $P(V_t|U_t, Y_1, \ldots, Y_t)$. For example, if the variables in $V_t$ are discrete and independent of each other given $U_t$, we can store for each variable the vector of parameters of the corresponding multinomial distribution (i.e., the probability of each value).

## 3 Probabilistic Relational Models

A *relational schema* is a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, where each class $C$ is associated with a set of *propositional attributes* $\mathcal{A}(C)$ and a set of *relational attributes* or *reference slots* $\mathcal{R}(C)$. The propositional attribute $A$ of class $C$ is denoted $C.A$, and its domain (assumed finite) is denoted $V(C.A)$. The relational attribute $R$ of $C$ is denoted $C.R$, and its domain is the power set $2^{C'}$ of a target class $C' \in \mathcal{C}$. In other words, $C.R$ is a set of objects belonging to some class $C'$.[1] For example, the *Aircraft* schema might be used to represent partially or completely assembled aircraft, with classes corresponding to different types of parts like metal sheets, nuts and bolts. The propositional attributes of a bolt might include its color, weight, and dimensions, and its relational attributes might include the nut it is attached to and the two metal sheets it is bolting. An *instantiation* of a schema is a set of objects, each object belonging to some class $C \in \mathcal{C}$, with all propositional and relational attributes of each object specified. For example, an instantiation of the aircraft schema might be a particular airplane, with all parts, their properties and their arrangement specified.

A *probabilistic relational model (PRM)* encodes a probability distribution over the set of all possible instantiations $I$ of a schema [Friedman *et al.*, 1999]. The *object skeleton* of an instantiation is the set of objects in it, with all attributes unspecified. The *relational skeleton* of an instantiation is the set of objects in it, with all relational attributes specified, and all propositional attributes unspecified. In the simplest case, the relational skeleton is assumed known, and the PRM specifies a probability distribution for each attribute $A$ of each class $C$. The parents of each attribute (i.e., the variables it depends on) can be other attributes of $C$, or attributes of

[1] $C.R$ can also be defined as a function from $C$ to $2^{C'}$, but we choose the simpler convention here.

classes that are related to $C$ by some slot chain. A slot chain is a composition of relational attributes. In general, it must be used together with an *aggregation function* that reduces a variable number of values to a single value. For example, a parent of an attribute of a bolt in the aircraft schema might be *avg(bolt.plate.nut.weight)*, the average weight of all the nuts on the metal plates that the bolt is attached to.

**Definition 1** A *probabilistic relational model (PRM)* $\Pi$ for a relational schema $\mathcal{S}$ is defined as follows. For each class $C$ and each propositional attribute $A \in \mathcal{A}(\mathcal{C})$, we have:

- A set of *parents* $Pa(C.A) = \{Pa_1, Pa_2, ..., Pa_l\}$, where each $Pa_i$ has the form $C.B$ or $\gamma(C.\tau.B)$, where $\tau$ is a slot chain and $\gamma()$ is an aggregation function.
- A *conditional probability model* for $P(C.A|Pa(C.A))$. □

Let $\mathcal{O}$ be the set of objects in the relational skeleton. The probability distribution over instantiations $I$ of $\mathcal{S}$ represented by the PRM is then

$$P(I) = \prod_{obj \in \mathcal{O}} \prod_{A \in \mathcal{A}(obj)} P(obj.A|Pa(obj.A))$$

A PRM and relational skeleton can thus be unrolled into a large Bayesian network with one variable for each attribute of each object in the skeleton.[2] Only PRMs that correspond to Bayesian networks without cycles are valid.

More generally, only the object skeleton might be known, in which case the PRM also needs to specify a distribution over the relational attributes [Getoor *et al.*, 2001]. In the aircraft domain, a PRM might specify a distribution over the state of assembly of an airplane, with probabilities for different faults (e.g., a bolt is loose, the wrong plates have been bolted, etc.).

## 4 Dynamic Probabilistic Relational Models

In this section we extend PRMs to modeling dynamic systems, the same way that DBNs extend Bayesian networks. We begin with the observation that a DBN can be viewed as a special case of a PRM, whose schema contains only one class $Z$ with propositional attributes $Z_1, \ldots, Z_d$ and a single relational attribute *previous*. There is one object $Z_t$ for each time slice, and the *previous* attribute connects it to the object in the previous time slice. Given a relational schema $\mathcal{S}$, we first extend each class $C$ with the relational attribute *C.previous*, with domain $C$. As before, we initially assume that the relational skeleton at each time slice is known. We can then define two-time-slice PRMs and dynamic PRMs as follows.

**Definition 2** A *two-time-slice PRM (2TPRM)* for a relational schema $\mathcal{S}$ is defined as follows. For each class $C$ and each propositional attribute $A \in \mathcal{A}(\mathcal{C})$, we have:

- A *set of parents* $Pa(C.A) = \{Pa_1, Pa_2, ..., Pa_l\}$, where each $Pa_i$ has the form $C.B$ or $f(C.\tau.B)$, where $\tau$ is a slot chain containing the attribute *previous* at most once, and $f()$ is an aggregation function.
- A *conditional probability model* for $P(C.A|Pa(C.A))$. □

[2]Plus auxiliary (deterministic) variables for the required aggregations, which we omit from the formula for simplicity.

**Definition 3** A *dynamic probabilistic relational model (DPRM)* for a relational schema $\mathcal{S}$ is a pair $(M_0, M_{\rightarrow})$, where $M_0$ is a PRM over $I_0$, representing the distribution $P_0$ over the initial instantiation of $\mathcal{S}$, and $M_{\rightarrow}$ is a 2TPRM representing the transition distribution $P(I_t|I_{t-1})$ connecting successive instantiations of $\mathcal{S}$. □

For any $T$, the distribution over $I_0, \ldots, I_T$ is then given by

$$P(I_0, \ldots, I_T) = P_0(I_0) \prod_{t=1}^{T} P(I_t|I_{t-1})$$

DPRMs are extended to the case where only the object skeleton for each time slice is known in the same way that PRMs are, by adding to Definition 2 a set of parents and conditional probability model for each relational attribute, where the parents can be in the same or the previous time slice. When the object skeleton is not known (e.g., if objects can appear and disappear over time), the 2TPRM includes in addition a Boolean existence variable for each possible object, again with parents from the same or the previous time slice.[3] As with DBNs, we may wish to distinguish between observed and unobserved attributes of objects. In addition, we can consider an *Action* class with a single attribute whose domain is the set of actions that can be performed by some agent (e.g., painting a metal plate, or bolting two plates together). The distribution over instantiations in a time slice can then depend on the action performed in that time slice. For example, the action *Bolt(Part1, Part2)* may with high probability produce *Part1.mate* = {*Part2*}, and with lower probability set *Part1.mate* to some other object of *Part2*'s class (i.e., be improperly performed, resulting in a fault).

Just as a PRM can be unrolled into a Bayesian network, so can a DPRM be unrolled into a DBN. (Note, however, that this DBN may in general contain different variables in different time slices.) In principle, we can perform inference on this DBN using particle filtering. However, the filter is likely to perform poorly, because for non-trivial DPRMs its state space will be huge. Not only will it contain one variable for each attribute of each object of each class, but relational attributes will in general have very large domains. We overcome this by adapting Rao-Blackwellisation to the relational setting. We make the following (strong) assumptions:

1. Relational attributes with unknown values do not appear anywhere in the DPRM as parents of unobserved attributes, or in their slot chains.
2. Each reference slot can be occupied by at most one object.

**Proposition 1** *Assumptions 1 and 2 imply that, given the propositional attributes and known relational attributes at times $t$ and $t-1$, the joint distribution of the unobserved relational attributes at time $t$ is a product of multinomials, one for each attribute.*

Notice also that, by Assumption 1, unobserved propositional attributes can be sampled without regard to unobserved relational ones. Rao-Blackwellisation can now be applied

[3]Notice that the attributes of nonexistent objects need not be specified, because by definition no attributes of any other objects can depend on them [Getoor *et al.*, 2001].

with $U_t$ as the propositional attributes of all objects and $V_t$ as their relational attributes. A Rao-Blackwellised particle is composed of sampled values for all propositional attributes of all objects, plus a probability vector for each relational attribute of each object. The vector element corresponding to $obj.R[i]$ is the probability that relation $R$ holds between $obj$ and the $i$th object of the target class, conditioned on the values of the propositional attributes in the particle, etc.

Rao-Blackwellising the relational attributes can vastly reduce the size of the state space which particle filtering needs to sample. However, if the relational skeleton contains a large number of objects and relations, storing and updating all the requisite probabilities can still become quite expensive. This can be ameliorated if context-specific independencies exist, i.e., if a relational attribute is independent of some propositional attributes given assignments of values to others [Boutilier *et al.*, 1996]. We can then replace the vector of probabilities with a tree structure whose leaves represent probabilities for entire sets of objects. More precisely, we define the *abstraction tree* data structure for a relational attribute $obj.R$ with target class $C'$ as follows. A node $\nu$ of the tree is composed of a probability $p$ and a logical expression $\phi$ over the propositional attributes of the schema. Let $\mathcal{O}_\nu(C')$ be the set of objects in $C'$ that satisfy the $\phi$'s of $\nu$ and all of $\nu$'s ancestors. Then $p \stackrel{\text{def}}{=} \sum_{obj' \in \mathcal{O}_\nu(C')} P(obj' \in (obj.R)_t \mid U_t)$. The root of an abstraction tree contains $\phi = true$. The children $\nu_i$ of a node $\nu$ contain expressions $\phi_i$ such that the $O_{\nu_i}(C')$ form a partition of $O_\nu(C')$. Each leaf of the tree stores a parametric distribution giving the probability that each object in the leaf is a member of $obj.R$, as a function of the object's propositional attributes. The probability that an arbitrary object $obj' \in C'$ is a member of $obj.R$ is found by starting at the root of the abstraction tree for $obj.R$, going to the child whose condition is satisfied by $obj'$, and so on recursively until a leaf is reached and the object's probability is read from the leaf distribution.

Initially, the abstraction tree consists only of the root, and as inference progresses it is gradually refined as dictated by the attributes that $C.R$ depends on. For example, suppose the first action to be performed is *Bolt(Part1, Part2)*, and with probability $p_f$ the action is performed incorrectly. The faulty action consists of attaching *Part1* to some other object of *Part2*'s class $C'$, with uniform probability over $C'$. Then two children $\nu_1$ and $\nu_2$ of the root of *Part1.mate*'s abstraction tree are created, with $\phi_1$ specifying the singleton set $\{Part2\}$ and $\phi_2$ its complement in $C'$, and with $p_1 = 1 - p_f$ and $p_2 = p_f$. The uniform distribution in leaf $\nu_2$ has a single parameter, the probability $p = p_f/(|C'| - 1)$ that a given object in it is attached to *Part1*. This takes $O(1)$ space to store and $O(1)$ time to update, as opposed to $O(|C'|)$. If objects with different attributes have different probabilities of being bolted to *Part1*, a node for each relevant combination of attributes is created. Thus, if $n_c$ is the number of such combinations, the storage and update time required for *Part1.mate* are $O(n_c)$ instead of $O(|C'|)$. By design, $n_c \leq (|C'|)$; in the worst case, the tree will have one leaf per element of $C'$. As we will see in the next section, the use of abstraction trees can greatly reduce the computational cost of Rao-Blackwellised particle filtering in DPRMs.

## 5 Experiments

In this section we study the application of DPRMs to fault detection in complex assembly plans. We use a modified version of the *Schedule World* domain from the AIPS-2000 Planning Competition.[4] The problem consists of generating a plan for assembly of objects with operations such as painting, polishing, etc. Each object has attributes such as surface type, color, hole size, etc. We add two relational operations to the domain: bolting and welding. We assume that actions may be faulty, with fault model described below. In our experiments, we first generate a plan using the FF planner [Hoffmann and Nebel, 2001]. We then monitor the plan's execution using particle filtering (PF), Rao-Blackwellised particle filtering (RBPF) and RBPF with abstraction trees.

We consider three classes of objects: *Plate*, *Bracket* and *Bolt*. *Plate* and *Bracket* have propositional attributes such as weight, shape, color, surface type, hole size and hole type, and relational attributes for the parts they are welded to and the bolts bolting them to other parts (e.g., *Plate73.bolt4* corresponds to the fourth bolt hole on plate 73). The *Bolt* class has propositional attributes such as size, type and weight. Propositional actions include painting, drilling and polishing, and change the propositional attributes of an object. The relational action *Bolt* sets a *bolt* attribute of a *Plate* or *Bracket* object to a *Bolt* object. The *Weld* action sets a *welded-to* attribute of a *Plate* or *Bracket* object to another *Plate* or *Bracket* object.

The fault model has a global parameter, the *fault probability* $p_f$. With probability $1 - p_f$, an action produces the intended effect. With probability $p_f$, one of several possible faults occurs. Propositional faults include a painting operation not being completed, the wrong color being used, the polish of an object being ruined, etc. The probability of different propositional faults depends on the properties of the object being acted on. Relational faults include bolting the wrong objects and welding the wrong objects. The probability of choosing a particular wrong object depends on its similarity to the intended object. Similarity depends on different propositional attributes for different actions and different classes of objects. Thus the probability of a particular wrong object being chosen is uniform across all objects with the same relevant attribute values.

The DPRM also includes the following observation model. There are two instances of each attribute: the true one, which is never observed, and the observed one, which is observed at selected time steps. Specifically, when an action is performed, all attributes of the objects involved in it are observed, and no others. Observations are noisy: with probability $1 - p_o$ the true value of the attribute is observed, and with probability $p_o$ an incorrect value is observed. Incorrect values for propositional observations are chosen uniformly. Incorrect values for relational observations are chosen with a probability that depends on the similarity of the incorrect object to the intended one.

Notice that, if the domain consisted exclusively of the propositional attributes and actions on them, exact inference might be possible; however, the dependence of relational attributes and their observations on the propositional attributes

[4]URL: http://www.cs.toronto.edu/aips2000

creates complex dependencies between these, making approximate inference necessary.

A natural measure of the accuracy of an approximate inference procedure is the K-L divergence between the distribution it predicts and the actual one [Cover and Thomas, 2001]. However, computing it requires performing exact inference, which for non-trivial DPRMs is infeasible. Thus we estimate the K-L divergence by sampling, as follows. Let $D(p||\hat{p})$ be the K-L divergence between the true distribution $p$ and its approximation $\hat{p}$, and let $\mathcal{X}$ be the domain over which the distribution is defined. Then

$$\begin{aligned} D(p||\hat{p}) &\stackrel{\text{def}}{=} \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{\hat{p}(x)} \\ &= \sum_{x \in \mathcal{X}} p(x) \log p(x) - \sum_{x \in \mathcal{X}} p(x) \log \hat{p}(x) \end{aligned}$$

The first term is simply the entropy of $X$, $H(X)$, and is a constant independent of the approximation method. Since we are mainly interested in measuring differences in performance between approximation methods, this term can be neglected. The K-L divergence can now be approximated in the usual way by taking $S$ samples from the true distribution:

$$\hat{D}_H(p||\hat{p}) = -\frac{1}{S} \sum_{i=1}^{S} \log \hat{p}(x_i)$$

where $\hat{p}(x_i)$ is the probability of the $i$th sample according to the approximation procedure, and the $H$ subscript indicates that the estimate of $D(p||\hat{p})$ is offset by $H(X)$. We thus evaluate the accuracy of PF and RBPF on a DPRM by generating $S = 10,000$ sequences of states and observations from the DPRM, passing the observations to the particle filter, inferring the marginal probability of the sampled value of each state variable at each step, plugging these values into the above formula, and averaging over all variables. Notice that $\hat{D}_H(p||\hat{p}) = \infty$ whenever a sampled value is not represented in any particle. The empirical estimates of the K-L divergence we obtain will be optimistic in the sense that the true K-L divergence may be infinity, but the estimated one will still be finite unless one of the values with zero predicted probability is sampled. This does not preclude a meaningful comparison between approximation methods, however, since on average the worse method should produce $\hat{D}_H(p||\hat{p}) = \infty$ earlier in the time sequence. We thus report both the average K-L divergence before it becomes infinity and the time step at which it becomes infinity, if any.

Figures 1 and 2 show the results of the experiments performed. The observation noise parameter $p_o$ was set to the same value as the fault probability $p_f$ throughout. One action is performed in each time step; thus the number of time steps is the length of the plan. The graphs show the K-L divergence of PF and RBPF at every 100th step (it is the same for RBPF with and without abstraction trees). Graphs are interrupted at the first point where the K-L divergence became infinite in any of the runs (once infinite, the K-L divergence never went back to being finite in any of the runs), and that point is labeled with the average time step at which the blow-up occurred. As can be seen, PF tends to diverge rapidly, while the

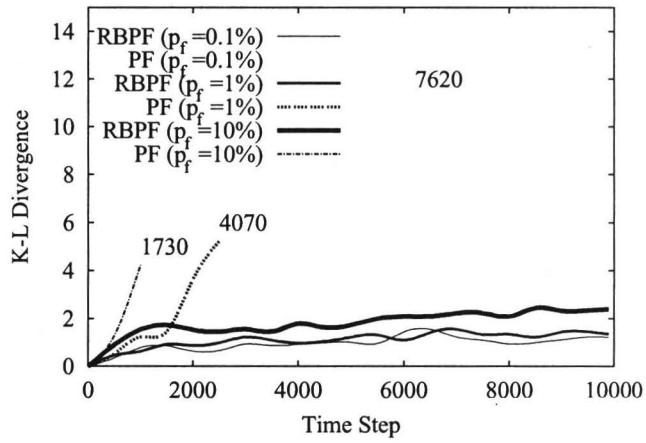


Figure 1: Comparison of RBPF (5000 particles) and PF (200,000 particles) for 1000 objects and varying fault probability.

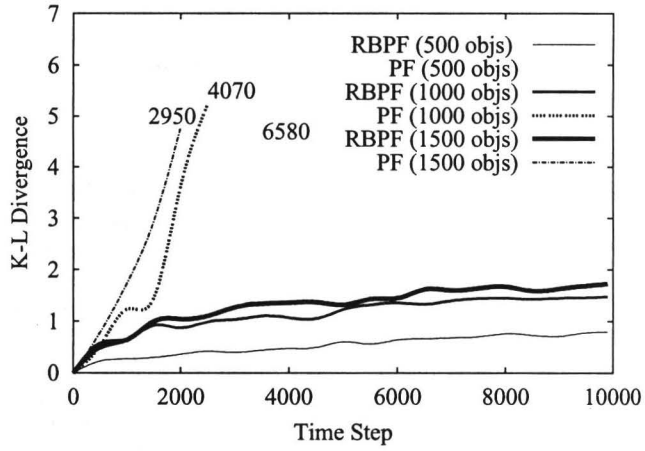


Figure 2: Comparison of RBPF (5000 particles) and PF (200,000 particles) for fault probability of 1% and varying number of objects.

K-L divergence of RBPF increases only very slowly, for all combinations of parameters tried. Abstraction trees reduced RBPF's time and memory by a factor of 30 to 70, and took on average six times longer and 11 times the memory of PF, per particle. However, note that we ran PF with 40 times more particles than RBPF. Thus, RBPF is using less time and memory than PF, and performing far better in accuracy.

We also ran all the experiments while measuring the K-L divergence of the full joint distribution of the state (as opposed to just the marginals). RBPF performed even better compared to PF in this case; the latter tends to blow up much sooner (e.g., from around step 4000 to less than 1000 for $p_f = 1\%$ and 1000 objects), while RBPF continues to degrade only very slowly.

## 6 Related Work

Dynamic object-oriented Bayesian networks (DOOBNs) [Friedman *et al.*, 1998] combine DBNs with OOBNs, a predecessor of PRMs. Unfortunately, no efficient inference

methods were proposed for DOOBNs, and they have not been evaluated experimentally. DPRMs can also be viewed as extending relational Markov models (RMMs) [Anderson *et al.*, 2002] and logical hidden Markov models (LOHMMs) [Kersting *et al.*, 2003] in the same way that DBNs extend HMMs. Downstream, DPRMs should be relevant to research on relational Markov decision processes (e.g., [Boutilier *et al.*, 2001]).

Particle filtering is currently a very active area of research [Doucet *et al.*, 2001]. In particular, the FastSLAM algorithm uses a tree structure to speed up RBPF with Gaussian variables [Montemerlo *et al.*, 2002]. Abstraction trees are also related to the abstraction hierarchies in RMMs [Anderson *et al.*, 2002] and to AD-trees [Moore and Lee, 1997]. An alternate method for efficient inference in DBNs that may also be useful in DPRMs was proposed by Boyen and Koller [1998] and combined with particle filtering by Ng et al. [2002]. Efficient inference in relational probabilistic models has been studied by Pasula and Russell [2001].

## 7 Conclusions and Future Work

This paper introduces dynamic probabilistic relational models (DPRMs), a representation that handles time-changing phenomena, relational structure and uncertainty in a principled manner. We develop efficient approximate inference methods for DPRMs, based on Rao-Blackwellisation of relational attributes and abstraction trees. The power of DPRMs and the scalability of these inference methods are illustrated by their application to monitoring assembly processes for fault detection.

Directions for future work include relaxing the assumptions made, further scaling up inference, formally studying the properties of abstraction trees, handling continuous variables, learning DPRMs, using them as a basis for relational MDPs, and applying them to increasingly complex real-world problems.

## 8 Acknowledgements

This work was partly supported by an NSF CAREER Award to the second author, by ONR grant N00014-02-1-0932, and by NASA grant NAG 2-1538. We are grateful to Mausam for helpful discussions.

## References

[Anderson *et al.*, 2002] C. Anderson, P. Domingos, and D. Weld. Relational Markov models and their application to adaptive Web navigation. In *Proc. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 143–152, Edmonton, Canada, 2002.

[Boutilier *et al.*, 1996] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 115–123, Portland, OR, 1996.

[Boutilier *et al.*, 2001] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. In *Proc. 17th Intl. Joint Conf. on Artificial Intelligence*, pages 690–697, Seattle, WA, 2001.

[Boyen and Koller, 1998] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proc. 14th Conf. on Uncertainty in Artificial Intelligence*, pages 33–42, Madison, WI, 1998.

[Cover and Thomas, 2001] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, New York, 2001.

[Dean and Kanazawa, 1989] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[Doucet *et al.*, 2001] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.

[Friedman *et al.*, 1998] N. Friedman, D. Koller, and A. Pfeffer. Structured representation of complex stochastic systems. In *Proc. 15th National Conf. on Artificial Intelligence*, pages 157–164, Madison, WI, 1998.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. 16th Intl. Joint Conf. on Artificial Intelligence*, pages 1300–1309, Stockholm, Sweden, 1999.

[Getoor *et al.*, 2001] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *Proc. 18th Intl. Conf. on Machine Learning*, pages 170–177, Williamstown, MA, 2001.

[Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Kersting *et al.*, 2003] K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden Markov models. In *Proc. 8th Pacific Symposium on Biocomputing*, Kauai, HI, 2003.

[Montemerlo *et al.*, 2002] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. 18th National Conf. on Artificial Intelligence*, pages 593–598, Edmonton, Canada, 2002.

[Moore and Lee, 1997] A. W. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, 1997.

[Murphy and Russell, 2001] K. Murphy and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 499–516. Springer, New York, 2001.

[Ng *et al.*, 2002] B. Ng, L. Peshkin, and A. Pfeffer. Factored particles for scalable monitoring. In *Proc. 18th Conf. on Uncertainty in Artificial Intelligence*, Edmonton, Canada, 2002. Morgan Kaufmann.

[Pasula and Russell, 2001] H. Pasula and S. J. Russell. Approximate inference for first-order probabilistic languages. In *Proc. 17th Intl. Joint Conf. on Artificial Intelligence*, pages 741–748, Seattle, WA, 2001.