UNIVERSITY OF CALIFORNIA,
IRVINE

Compiling Probabilistic Conformant Planning into Mixed Dynamic Bayesian Network

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Science

by

Junkyu Lee

Thesis Committee:
Professor Rina Dechter, Chair
Associate Professor Alexander Ihler
Professor Eric Mjolsness

2014

DEDICATION

To my beloved wife Younglim,
my sister,
my parents,
and my grand parents.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

ACKNOWLEDGMENTS

First I would like to thank Professor Rina Decther for her guidance and support during my MS program. It is truly my greatest honor and pleasure to have her as the Chair of my committee. I would like to thank other faculty members – Professor Alexander Iher, and Eric Mjolsness – for being my committee and their comments.

I am indebted to all those who helped me finish this thesis. I would like to thank Professor Hector Geffner for his kind comments about this work. I would also like to thank our research group members, Natalia Flerova, and Silu Yang. Especially, many thanks to William Lam and Radu Marinescu for their collaboration and advice. This thesis would not have been possible without their support.

Finally, I would like to thank my wife, my sister, and my parents for their endless love and support.

ABSTRACT OF THE THESIS

Compiling Probabilistic Conformant Planning into Mixed Dynamic Bayesian Network

By

Junkyu Lee

Master of Science in Computer Science

University of California, Irvine, 2014

Professor Rina Dechter, Chair

Probabilistic conformant planning is a task of finding a plan that achieves the goal without sensing, where the outcome of an action is probabilistic and the initial state is uncertain. In this thesis, we formulate the probabilistic conformant planning as marginal Maximum A Posteriori (MAP) probabilistic inference based on the finite horizon state transition model. In practice, most of the planning problems are expressed in Probabilistic Planning Domain Definition Languag. Therefore, we developed a translation that reads a PPDDL instance and compiles the instance into a graphical model to provide a planning problem to existing marginal MAP solvers. The compilation is based on SAT encoding of planning problems, and the encoding is extended from the linear encodings used to solve classical planning problems by SAT solvers. The graphical model is obtained by converting CNF clauses into a mixed network, where the probabilistic state transitions are compiled as Bayesian network and deterministic constraints are compiled as an auxiliary network. We performed empirical evaluation to compare marginal MAP algorithms and Probabilistic-FF planner. The experiment results show that marginal MAP algorithms were able to solve selected problem domains.

Chapter 1

Introduction

Planning is a process of selecting and organizing actions to achieve desired goal. The result of execution of an action can be either deterministic or non-deterministic, and the state of the world can be either observed, partially observed, or non-observed based on the environment. Classical planning refers to the category in which the effect of an action is deterministic and an agent can fully observe the initial state. On the other hand, a probabilistic planning agent should expect that the outcome of an action will be probabilistic or non-deterministic. We can further divide the probabilistic planning category based on the observability of the states: 1. fully observable probabilistic planning, a.k.a., FOMDP (Fully Observable Markov Decision Process), 2. partially observable probabilistic planning, a.k.a., POMDP (Partially Observable Markov Deicion Process), and 3. non-observable probabilistic planning, a.k.a., NOMDP (Non-Observable Markov Decision Process).

Conformant planning is equivalent to non-observable probabilistic planning, where an agent has uncertainty in both the inital state and state transitions, and tries to achieve the goal with certainty or some threshold value which indicates the probability of success.

Nondeterminism of the initial state or state transitions can be provided by a probability distribution over possible states, which is so called belief state. In practice, a belief state can be either specified quantitatively with a probability distribution or qualitatively with a special clause like "one-of" or "unknown". Whereas the reasoning about belief states can be reduced to probabilistic inference in the first case, the second case looks closer to classical setting.

We consider probabilistic conformant planning problem P as the task of generating a sequence of actions achieving the goal without sensing. In other words, a planner should find a sequence of actions in the quantitative belief state space resulting from uncertainty in both the initial state and action effects. The query to the planner can be formulated in two different ways: $\langle P, \theta \rangle$, which asks for a plan achieving the goal with probability that exceeds a threshold θ, and $\langle P, L \rangle$, which fixes the length of the action sequence L and asks for a plan with maximum probability. We define an optimal probabilistic conformant planner as a planner which always finds the shortest length plan that exceeds a threshold. Note that a planner which returns a plan from the query $\langle P, L \rangle$ can be transformed into an optimal planner by increasing the L incrementally.

Probabilistic conformant planning has been adressed less popular than the conformant planning with qualitative belief states. Indeed, the International Planning Competition offers all the planning problems except the probabilistic conformant planning. There are a few planners that can handle the probabilistic conformant planning problem in the literature. COMPLAN [10] and CPPlan [11] answer to the query $\langle P, L \rangle$, and Probabilistic Fast Forward [7] returns some satisfying plan to the query $\langle P, \theta \rangle$.

In this thesis, we are interested in solving the probabilistic conformant planning problem via a general purpose probabilistic infernece engine. Specifically, the probabilistic conformant planning problem is compiled as a marginal MAP inference problem and we provided the translated problems to recently developed marginal MAP algorithms [15]. Thus, the

overall approach is parallel to that of plannig as SAT, where a PDDL instance is compiled to CNF clauses and a SAT solver solves the problem. The contribution of this thesis is a new compilation method that translate existing probabilistic planning problems into graphical models, so that a probabilistic conformant planning problem can be solved by general purpose probabilistic inference engine.

The rest of this thesis is organized as follows. Following sections in Chapter 1 provides preliminaries, Chapter 2 presents the method of compiling a planning problem into a graphical model. In particular, it covers formulation of the marginal MAP query that solves probabilistic conformant planning problems, planning problem formalisms, planning domain languagues, and the process of compliing SAT encodings into a graphical model. Chapter 3 shows the empirical evaluations of applying marginal MAP algorithms and compares marignal MAP algorithms with existing planners. Finally, Chapter 4 concludes the thesis and proposes future directions.

1.1 Probabilistic Conformant Planning

The planning problem can be described as a state transition model with flat state/action representation $\langle S, \mathbf{s_i}, \mathbf{s_g}, A, T \rangle$, where the $S = \{\mathbf{s_0}, \mathbf{s_1}, \cdots, \mathbf{s_k}\}$ is a set of world states, each of them represents a mutually disjoint configuration in the world, the $\mathbf{s_i} \in S$ and the $\mathbf{s_g} \in S$ is the inital state and the goal state, the $A = \{\mathbf{a_0}, \mathbf{a_1}, \cdots, \mathbf{a_l}\}$ is a set of actions, and the T is the state transition function represented as a mapping either $T : S \times A \times S$, if the actions were deterministic, or $T : S \times A \times S \rightarrow [0, 1]$, if the actions were probabilistic. In factored state representation, each state is represented by state variables, where each state variable takes finite values from its domain. For example, a state $\mathbf{s}$ can be represented by n state variables $(s_0, s_1, \cdots s_n)$, and each s_j takes value from $Dom(s_j)$. Thus, the set of states S is represented as a cartesian product $Dom(s_0) \times Dom(s_1) \times \cdots, \times Dom(s_n)$ in

factored state representation. Similarly, actions can be factored with action variables. In practice, planning problems are represented in planning domain languages rather than the state transition model. Planning domain languages and planning problem formalisms are explained in Chapter 2. Now, we define the probabilistic conformant planning as follows.

DEFINITION **1.** ***Probabilistic Conformant Planning*** P is given by $\langle S, \mathbf{b_i}, \mathbf{s_G}, A, T \rangle$, where the $\mathbf{b_i}$ is the belief states of the initial states, and the $\mathbf{s_G} = \{\mathbf{s_{g_0}}, \cdots, \mathbf{s_{g_k}}\}$ is the set of goals in which the plan must satisfy one of the goal states, and the T is probabilistic state transition function, $T : S \times A \times S \rightarrow [0, 1]$. The belief state $\mathbf{b}$ is a probability distribution over states $\mathbf{b} : S \rightarrow [0, 1]$. The task is to find a sequence of action that achieves the goal certainly.

DEFINITION **2.** ***Finite Horizon Probabilistic Conformant Planning*** $\langle P, L \rangle$ is a probabilistic conformant planning problem augmented with the length L of the action sequence. The task is to find length L plan that achieves the goal with maximum probability.

DEFINITION **3.** ***Probabilitic Conformant Planning with Threshold*** $\langle P, \theta \rangle$ is a probabilistic conformant planning problem augmented with the threshold θ. The task is to find any plan with probability of achieving the goal higher than the θ.

DEFINITION **4.** ***Optimal Probabilistic Conformant Plan*** is a plan that achieves maximum probability of success given fixed plan length or the minimal length plan that exceeds the threshold.

1.2 Graphical Models and Probabilistic Inference Queries

Graphical model is a powerful machinery for knowledge representation and reasoning. The graph-based representation allows structure of a problem to be encoded compactly, where the nodes in a graph correspond to the variables in the problem, and two nodes are

connected by an edge if there exists some relation between them, i.e., the primal graph of a graphical model.

DEFINITION **5.** ***graphical model*** is a tuple $\mathcal{R} = \langle X, D, F, \bigotimes \rangle$, where the $X = \{x_0, \cdots, x_n\}$ is a set of variables, the $D = \{Dom(x_0), \cdots, Dom(x_n)\}$ is a set of domains of variables, $x_i \in Dom(x_i)$, the $F = \{F_0, \cdots, F_m\}$ is a set of functions defined over a set $S = \{S_0, \cdots, S_m\}$ of a scope $S_j \subseteq X$ for the function F_j, and the $\bigotimes$ is a combination operator such as the product $\prod$, the sum $\sum$, or the join $\bowtie$ operator depending on the problem formulation.

DEFINITION **6.** ***primal graph*** $G = \langle V, E \rangle$ of a graphical model $\mathcal{R}$ has a set of nodes $v_i \in V$ from the variables $x_i \in X$, and a set of edges $e_{ij} \in E$ if two variables x_i and x_j are members of some scope of a function F_k, $\{x_i, x_j\} \subseteq S_k$.

Popular frameworks for graphical models include *Bayesian networks*, *Markov networks*, *Constraint networks*, and *Mixed networks* which explicitly distinguish between probabilistic information and deterministic constraints.

DEFINITION **7.** ***Bayesian network*** is a graphical model $\mathcal{R}$ represented by a directed acyclic graph G, where the X is a set of multi-valued random variables, and each function $F_i \in F$ is a conditional probability table, $F_i(S_i) \equiv Pr(x_i|par(x_i))$. The $par(x_i)$ are the parent nodes of the variable x_i in G. The joint probability distribution over variable X is the product of all conditional probability distributions, $Pr(X) = \prod_{x_i \in X} Pr(x_i|par(x_i))$.

DEFINITION **8.** ***Markov network*** is a graphical model $\mathcal{R}$ represented by a undirected graph G, where the X is a set of multi-valued variables, and each function $F_i \in F$ is a local potential function defined over a subset of variables S_i, $F_i(S_i) \equiv \phi_i(S_i)$. Whereas the joint probability distribution of the Bayesian network has to be normalized to 1, the global potential function $\phi(X) = \prod_i \phi_i(S_i)$ is not necessarily normalized.

If a function in a graphical model is a deterministic constraint or a functional relation, all

the values of the function are either 1 or 0. For example, a CNF (conjunctive normal form) clause in propostional satisfiability (SAT) can be encoded as a factor table, where the scope of the factor is boolean variables in the clause, each row of the table is a truth assignment to all variables in the clause, and the value of the row is either 1 if the assignment is a model of the clause, or 0 if the assignment does not satisfy the clause. Practical problem domains such as probabilistic planning problems carry a large amount of symmetric and deterministic constraints on the top of causal relations which are normally represented as Bayesian networks. Hybrid processing of probabilistic information that are quantified by conditional probability table and deterministic information that are represented as a set of boolean clauses or constraints leverages computational efficiency of inference algorithms as well as compact compilation of knowledge [5, 17]. *Mixed network* is the overarching framework for representation and inference over mixture of a probabilistic graphical model (belief network) and a deterministic graphical model (constraint network).

DEFINITION **9.** ***Constraint network*** is a graphical model $\mathcal{R} = \langle X, D, F \rangle$, where the functions are constraints $F_i \equiv C_i = (S_i, R_i)$. Each constraint C_i is defined over its scope S_i with the relations R_i which denotes allowed combinations of values to the variables in S_i.

DEFINITION **10.** ***Mixed network*** can be defined given a belief network $\mathcal{B}$ and a constraint network $\mathcal{C}$, where the $\mathcal{B} = \langle X_\mathcal{B}, D_\mathcal{B}, P_\mathcal{B} \rangle$ defines the joint probability distribution $Pr_\mathcal{B}(X_\mathcal{B})$ over the variables $X_\mathcal{B}$, and the $\mathcal{C} = \langle X_\mathcal{C}, D_\mathcal{C}, C_\mathcal{C} \rangle$ is a constraint network that defines a set of solutions $\rho(X_\mathcal{C})$ which satisfies the constraints $C_\mathcal{C}$ over the variables $X_\mathcal{C}$. The mixed network is a graphical model $\mathcal{M} = \langle X_\mathcal{M}, D_\mathcal{M}, P_\mathcal{M} \rangle$, where the $X_\mathcal{M} = X_\mathcal{B} \cup X_\mathcal{C}$, the $D_\mathcal{M} = D_\mathcal{B} \cup D_\mathcal{C}$, the $P_\mathcal{M} = P_\mathcal{B} \cup C_\mathcal{C}$, and the joint probability distribution $Pr_\mathcal{M}(X)$ in conjunction with $Pr_\mathcal{B}(X_\mathcal{B})$ and $\rho(X_\mathcal{C})$ is defined as follows.

$$Pr_\mathcal{M}(\bar{x}) = \begin{cases} Pr_\mathcal{B}(\bar{x}), & \text{if } \bar{x} \in \rho(X_\mathcal{C}) \\ 0, & \text{otherwise.} \end{cases} \tag{1.1}$$

The probabilistic inference queries over the a belief network contains the posterior marginal, the probability of evidence, the most probable explanation (MPE), and the marginal maximum a posteriori probability (MAP). The above queries can be extended straight-forwardly in mixed networks [4].

Given a belief network $\mathcal{B} = \langle X, D, P \rangle$ and a set of evidence variables E in which variables are already assigned to some value as $\bar{e}$, we define typical probabilistic inference queries, where the P is either a set of conditional probability tables for Bayesian networks or a set of local potential functions for Markov networks, and the product of all elemements in P is the joint probability distribution $Pr_{\mathcal{B}}(X)$.

DEFINITION **11.** ***Posterior marginal*** is the marginal probability of a variable x_i given a set of evidence, $Pr_{\mathcal{B}}(x_i|\bar{e}) = \sum_{X-x_i} \prod_j P(x_j|par(x_j), \bar{e})$.

DEFINITION **12.** ***Probabiliy of evidence*** is the marginal probability of evidence, $Pr_{\mathcal{B}}(\bar{e}) = \sum_X \prod_j P(x_j|par(x_j), \bar{e})$.

DEFINITION **13.** ***MPE*** is an optimization task of finding an assinment of variables that maximize the probability of evidence, $\bar{\mathbf{x}} = \text{argmax}_X \prod_j P(x_j|par(x_j), \bar{e})$. The MPE value is $Pr_{\mathcal{B}}(\bar{\mathbf{x}})$.

DEFINITION **14.** ***Marginal MAP*** is also an optimization task of finding a partial assignment of hypothesized variables $A = \{a_0, \cdots, a_k\} \subseteq X$ after marginalizing the rest of the variables, $\bar{\mathbf{a}} = \text{argmax}_A \sum_{X-A} P(X|\bar{e}) = \text{argmax}_A \sum_{X-A} \prod_j (x_j|par(x_j), \bar{e})$.

The key component of this thesis are the mixed network and marginal MAP inference since a planning problem will be compiled into a mixed network and such a problem will be solved by the marginal MAP inference.

Chapter 2

Compiling Probabilistic Conformant Planning into Mixed DBN

This chapter explains the process of compiling a probabilistic conformant planning problem into a mixed network. The first section derives the formulation for solving a probabilistic conformant planning with a fixed time horizon by the marginal MAP inference, and the second section rewviews planning problem formalisms and planning domain specific languages. Finally, we explain detail process of compilation.

2.1 Conformant Planning as Marignal MAP

We formulate probabilistic conformant planning (PCP) with fixed time horizon, $\langle P, L \rangle$ as the marginal MAP query by use of state variables, action variables, and the joint conditional distribution conditioned on the initial and goal state variables and the action variables.

In Chapter 1, a planning problem was defined as a state transition model with factored representation. The state transition model of a planning problem with time indices $t \in$

$\{0, 1, \cdots, L\}$ is rewritten as follows.

DEFINITION **15.** ***PCP with fixed time horizon L*** is a triplet $\langle S, A, T \rangle$,

- $S = \{\mathbf{s^0}, \mathbf{s^1}, \cdots, \mathbf{s^L}\}$ is a set of $\mathbf{s^t}$ from time index 0 to L. $\mathbf{s^t} = \{s_0^t, \cdots, s_n^t\}$ is a state variable vector at the t-th time, where $n+1$ is the total number of state variables that fully encode all the states at t-th time.
- $A = \{\mathbf{a^0}, \mathbf{a^1}, \cdots, \mathbf{a^{L-1}}\}$ is a set of $\mathbf{a^t}$ from time index 0 to $L-1$. $\mathbf{a^t} = \{a_0^t, \cdots, a_m^t\}$ is an action variable vector at the t-th time, where $m+1$ is the total number of action variables that fully encode all the actions at the t-th time.
- T is a Markovian state transition function, $\forall t, T : \mathbf{s^t} \times \mathbf{a^t} \times \mathbf{s^{t+1}} \rightarrow [0, 1]$.

Note that the above definition is equivalent to that of the finite horizon MDP formulation with the maximum time length L+1. The initial state is $\mathbf{s^0}$ and the goal state is $\mathbf{s^L}$, and the state transition function $T(\mathbf{s^t}, \mathbf{s^{t+1}}, \mathbf{a^t})$ from time t to $t+1$ is a conditional distribution $Pr(\mathbf{s^{t+1}}|\mathbf{s^t}, \mathbf{a^t})$.

The probabilistic conformant planning task is to find a sequence of actions that maximizes the probability of achieving the goal given the fixed length L of the plan. Since the state transition function is Markovian, the joint probability distribution over the set of state variables and a sequence of actions $\{\mathbf{a^0}, \mathbf{a^1}, \cdots, \mathbf{a^{L-1}}\}$ can be factored as follows.

$$
\begin{aligned}
Pr(\mathbf{s^0}..\mathbf{s^L}|\mathbf{a^0}..\mathbf{a^{L-1}}) &= \prod_{i=0..L} Pr(\mathbf{s^i}|\mathbf{s^0}..\mathbf{s^{i-1}}, \mathbf{a^0}..\mathbf{a^{L-1}}) && (2.1)\\
&= \prod_{i=0..L} Pr(\mathbf{s^i}|\mathbf{s^{i-1}}, \mathbf{a^{i-1}}) && (2.2)\\
&= Pr(\mathbf{s^0}) Pr(\mathbf{s^L}|\mathbf{s^{L-1}}, \mathbf{a^{L-1}}) \prod_{i=1..L-1} Pr(\mathbf{s^i}|\mathbf{s^{i-1}}, \mathbf{a^{i-1}}) && (2.3)
\end{aligned}
$$

If the initial state was $\mathbf{s_I}$ and the goal state was $\mathbf{s_G}$, the above probability distribution can

be rewritten as,

$$Pr(\mathbf{s^0}..\mathbf{s^L}|\mathbf{s^0}=\mathbf{s_I},\mathbf{s^L}=\mathbf{s_G},\mathbf{a^0}..\mathbf{a^{L-1}}) \tag{2.4}$$

$$= \quad Pr(\mathbf{s^0}=\mathbf{s_I})Pr(\mathbf{s^L}|\mathbf{s^L}=\mathbf{s_G},\mathbf{s^{L-1}},\mathbf{a^{L-1}})\prod_{i=1..L-1}Pr(\mathbf{s^i}|\mathbf{s^{i-1}},\mathbf{a^{i-1}}) \tag{2.5}$$

where $Pr(\mathbf{s^0}=\mathbf{s_I})$ is the initial belief state, and the last state is implicitly constrained to be the goal state, $Pr(\mathbf{s^L}|\mathbf{s^L}=\mathbf{s_G},\mathbf{s^{L-1}},\mathbf{a^{L-1}})=0$, if $\mathbf{s^L}\neq\mathbf{s_G}$. Finally, the probabilistic conformant planning task is equivalent to the marginal MAP query in which the action variables correspond to the hypothesis variables $(\mathbf{a^0}..\mathbf{a^{L-1}})$.

$$(\mathbf{a^0}..\mathbf{a^{L-1}})=\arg\max_{(\mathbf{a^0}..\mathbf{a^{L-1}})}\sum_{\mathbf{s^i}\in S}Pr(\mathbf{s^1}..\mathbf{s^{L-1}}|\mathbf{s^0}=\mathbf{s_I},\mathbf{s^L}=\mathbf{s_G},\mathbf{a^0}..\mathbf{a^{L-1}}) \tag{2.6}$$

If multiple goal states were allowed, the constraint on the last state variable can be extended as $Pr(\mathbf{s^L}|\mathbf{s^L}\in\{\mathbf{s_G}\},\mathbf{s^{L-1}},\mathbf{a^{L-1}})=0$, if $\mathbf{s^L}\notin\{\mathbf{s_G}\}$.

2.2 Planning Problem Formalisms and Languages

In the previous section, probabilistic conformant planning was formulated as the marginal MAP inference based on the state transition model, a.k.a., finite horizon MDP (FH-MDP). However, practical planning problems are usually formulated by STRIP like formalisms and expressed in the planning domain specific languages. Therefore, it is desired to design a translation method that converts the planning domain specific languages to the FH-MDP formulation. We assume that the result of the translation will be represented by tabular forms i.e., *UAI format* which has been widely used in UAI (Uncertainty in Artificial Intelligence) community [1].

The STRIPS (Stanford Research Institute Problem Solver) formalism refers to the original

proposal for representing classical planning problems in [8].

DEFINITION **16.** ***Classical Propositional STRIPS*** formalism is a quadraple $\langle P, O, I, G\rangle$, where the P is a set of propositional atoms, the O is a set of operators (actions), the I is a list of positive atoms that must be true at the initial state, the G is a list of positive atoms that must be true at the goal state. Each operator $o \in O$ contains three lists of positive atoms $\langle pre(o), add(o), del(o)\rangle$, where the $pre(o)$ is a precocondition list that must be satisfied to exectue the operator o, the $add(o)$ is an add list that will be true after executing the operator o, and the $del(o)$ is a delete list that will be false after executing the operaotr o.

The STRIPS planning system maintains positive propositional atoms with closed world assumption. The semantic of applying an operator at a state can be described as follows. A planner maintains a list of positive atoms that is true at each state s. If $pre(o) \subseteq s$, executing the operator o will transform the current list of atoms of the state s to $result(s, o) = s \cup add(o) - del(o)$. On the other hand, if $pre(o) \nsubseteq s$, executing the operator o is not defined. The plan is a sequence of operator $\{o_0, \cdots, o_n\}$ that satisfies $pre(o) \subseteq I$, $s_1 = result(I, o_0)$, $pre(o_i) \subseteq s_i$, and $G \subset result(o_n)$.

The orginal CPS formalism did not allow negated atoms in its precondition list and the goal list. Relaxing such restrictions gives the formalism, Propositional STRIPS with Negative goals (PSN).

Action Description Language (ADL) is another formulation for the classical planning problems. ADL features are highlighted and compared to CPN in Table 2.1, and we consult [21] for deeper descrition about ADL. The notable difference between the STRIPS and the ADL is that the ADL introduces variable terms for expressing states, and quantifiers, equality predicates, and types for such variables. In addition, disjunction is also allowed in condition expressions. Researchers proposed various planning languages that are inherited from STRIPS and ADL by incorporating the first order language constructs: lifted

	STRIPS	ADL
States	Conjunction of positive literals	Conjunction of literals
Goal state	Only positive ground literals	Allow quantified variables
Goal expression	Conjunction	Allow Conjunction and disjunction
Operator expression	Conjunction	Allow Conditional effects
Unmentioned literals	Closed world assumption	Open world assumption
Equality predicates	No equality	Allow equality predicates for terms
Types	No types	Allow types for variables

Table 2.1: Comparison between STRIPS and ADL

predicates, lifted action operators, a.k.a., action schemata, types for the terms, equality predicates, quantifiers, disjunction, and etc.

The Planning Domain Definition Language (PDDL) is de facto standard language for classical planning problems. The first version of PDDL was published in 1998 as the official language for the first international planning competition, and it has been subsequently extended by various language constructs, for example, numeric states, plan metrics, durative actions, soft constraints, just to name a few. In this thesis, the scope of PDDL is limited to level 1, the usual STRIPS and ADL planning. In PDDL, a planning problem instance is decribed by the *domain* definition part and the *problem* definition part. The *domain* definition part specifies a planning problem in first order syntax that can be reused with various *problem* definitions which only specify instance specific information such as ground objects, the ground initial state, and the ground goal states. A compact summary of PDDL syntax is given in Figure 2.1. Note that the following syntax is informal, and the exact PDDL syntax should be consulted in the language specification[18]. As mentioned, the core language constructs of PDDL are first order predicates and action schemata. Notable features of PDDL are: 1. a condition expression in an action schema or in a conditional effect allows an arbitrary function free first order sentence, 2. an effect, the result of executing an action, can be nested to arbitarry depth by conditional effects and conjunction of effects, 3. the goal is also allowed to be an arbitrary function free first order sentence, and 4. the default value of unspecified predicates are true, a.k.a., closed world assumption.

```
<domain>              ::= <predictes> <actions>
<predicates>          ::= list of <predicate>
<predicate>           ::= (<name> <list of variables>*)
<actions>             ::= list of <action>
<action>              ::= (<name> <list of variables>* <action body>)
<action body>         ::= [<precondition>] [<effect>]
<precondition>        ::= <ground expression>
<ground expression>   ::= <predicate> <list of variables>* |
                          equality on two predicates |
                          negation of a precondition |
                          existentially quantified precondition |
                          universally quantified precondition |
                          conjunction of preconditions |
                          disjunction of preconditions |
<effect>              ::= <simple effect> |
                          <conditional effect> |
                          conjunction of effects
<simple effect>       ::= predicate literal
<conditional effect>  ::= when <precondition> <effect>
<problem>             ::= <ground terms> <init state> <goal>
<ground terms>        ::= list of ground objects
<init state>          ::= conjunction of ground predicates
<goal>                ::= <ground expression>
```

Figure 2.1: Informal PDDL Syntax with core parts of the language.

Probabilistic Planning Domain Definition Language (PPDDL 1.0) [24] is a probabilistic extension from PDDL 2.1, and it was introduced at the 4th IPC probabilistic planning track. It extended PDDL by adding probabilistic effects, and rewards to express MDP planning. The MDP planning is out of our scope in this thesis, so only the syntax for the probabilistic effect is shown in Figure 2.2.

The probabilistic effect is a list of effects quantified by probability values. If the sum of

```
<effect>          ::= <simple effect> |
                      <conditional effect> |
                      <prob. effect> |
                      conjunction of effects
<prob. effect>    ::= list of pairs (p, <effect>)
```

Figure 2.2: Extending PDDL to PPDDL by Adding Probabilistic Effect

values is less than 1.0, the PPDDL semantics assumes that there exits an additional probabilistic outcome that is null effect with the deficient probability value. The effect syntax in PPDDL also allows arbitrary nested effects, but such nested effects can be flattened by normal forms proposed by [20].

2.3 Compiling PPDDL into Mixed DBN

The process of compiling a PPDDL instance into a mixed Dynamic Bayesian Network (DBN) is explained in this section. We obtain a DBN by replicating the structure of 2 time stage Dynamic Bayesian network (2TDBN) up to desired time horizon since the probabilistic inference query for the probabilistic conformant planning was formulated based on the state transition model with a fixed time length. The term *mixed* emphasizes that the result of the compilation is DBN augmented with deterministic constraints encoded by CNF clauses.

The overall process of compiling a PPDDL instance into a mixed DBN is illustrated in Figure 2.3. The first step is to encode the original PPDDL instance as CNF clauses that separte the planning constraints and probabilistic state transitions. In practice, a PPDDL instance contains a lot of functional relations such as the frame axiom, the mutual exclusivity axiom, and condition expressions that must be satisfied to activate the effect of some action. If precondition of an action or condition of a conditional effect was not satisfied, the effect of executing the action must be the null effect. SAT clauses are useful for identifying such constraints on top of probabilistic state transitions triggered by an instantiation of some action variable. In the PPDDL 1.0 specification, it is shown that a 2TDBN can be obtained for an action without, but the 2TDBN did not combine all the possible state transitions that can be instantiated by all possible combinations of action variables. One could compile such many 2TDBNs into a single global 2TDBN by stacking them into a

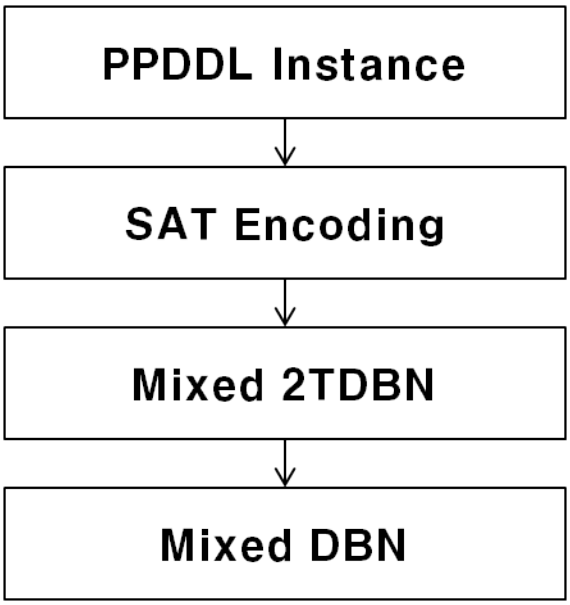


Figure 2.3: Overall Process of Compiling PPDDL into Mixed DBN. The first step is encoding CNF clauses that extract deterministic constraints in a PPDDL instance, the second step is compiling a mixed 2TDBN from the probabilistic effects and CNF clauses, and the last step is extending 2TDBN to desired time horizon and encodes the initial belief state and the goal constraints.

single 2TDBN by introducing action variables [2]. However, such a naive approach will result in exponentially huge conditional probability tables that does not separate deterministic constraints inside the planning problem. Thus, our contribution is that we developed a compilation process that explicitly separates and encodes deterministic constraints as a mixed 2TDBN. The final mixed DBN can be obtained in a staright forward manner by replicating the mixed 2TDBN up to desired time horizon.

2.3.1 SAT Encoding for PPDDL

The SAT encoding for a PPDDL instance that we introduce is a direct extension of the linear encoding for a PDDL instance[12]. Given a ground PPDDL instance, we have a set of ground predicates S and ground action schemata A. We introduce a boolean state variable for each ground predicate $s_i \in S$, and a boolean action variable for each action schemata

$a_j \in A$. A ground action schema a_j consists of an action precondition ϕ, where the ϕ can be regarded as a CNF clause over the state variables, and a nested ground effect e which can be flattend to one of the normal forms in [20]. Without loss of generality, we assume that an effect e is expressed by the unary nondeterminism normal form, where a nested effect is reduced to a list of conjunctions of conditional effects, each of them is annotated with a probability value. Thus, an effect can be written as $e = ((p_0, c_0), (p_1, c_1), \cdots, (p_n, c_n))$, where the sum of the probability is normalized to 1, $\sum_{i=0..n} p_i = 1$, and each c_i is conjunction of conditional effects, $\wedge_j(\phi_j \rhd \omega_j)$, with CNF clauses ϕ_j and conjunction of state variables ω_j. Note that conjunction of state variables is deterministic simple effect. For simplicity, the time indices are dropped, and CNF clauses for single time stage transition is described as follows.

- For each ground predicate/action, introduce a boolean state/action variable s_i/a_i.
- For each action a_i, introduce a multi-valued effect variable e_{a_i} which has $n+1$ values if the effect had n outcomes. The first value of an effect variable e_{a_i} is *no-op*, which means that the result of the effect will be null effect, and the rest of the values refer to conditional effects c_j defined earlier.
- For each ground action a_i, let ϕ_i be a CNF clause for a action precondition, then $a_i \wedge \phi_i \Leftrightarrow (e_{a_i} \neq \text{no-op})$, where the $(e_{a_i} = v)$ is an equality predicate that is true if the value of the multi-valued variable e_{a_i} equals v.
- For each state variable s_i, we introduce two auxiliary boolean variables for state transition, +s_i and -s_i. The +s_i is true if any one of the action that could add the state variable s_i at the next time stage was executed. Similary the -s_i is true if disjunction of actions that could delete the state variable s_i was true. Note that the v-th value of an effect variable e_{a_i} refers to a conjunction of deterministic conditional effects $\wedge_j(\phi_j \rhd \omega_j)$. Assuming that the v-th outcome of an effect e_{a_i} was occured, the state

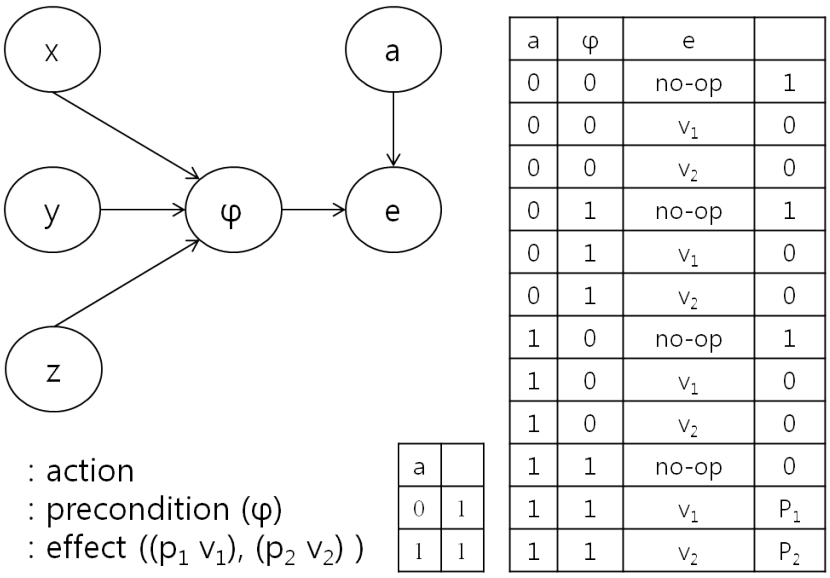
variable s_i must be true if the s_i appeared as a positive literal in one of the simple effects ω_j of the c_i, and if the condition ϕ_j was satisfied by current state variables. For simplicity, we illustrate a CNF cluase for the case where the conditions ϕ_j are always true, for all e_{a_i} s.t. $\exists v \in Dom(e_{a_i})$ s.t. $s_i \in add(e_{a_i} = v)$, $\vee_{e_{a_i}}(e_{a_i} = v) \Leftrightarrow$ +s_i. The $add(e_{a_i} = v)$ is the add list of the simple effect of the v-th outcome of the effect. The delete effect for the -s_i can be defined in the same way.

- Since our scope is limited to the linear encoding, only single outcome can be activated at each time stage. Hence, if $s_i \in add(e_{a_i} = v_i)$ and $s_i \in add(e_{a_j} = v_j)$, $\forall_{a_i,a_j}(e_{a_i} = v_i) \wedge (e_{a_j} = v_j) \rightarrow \neg$+$s_i$. We have the same clause for the delete effect, if $s_i \in del(e_{a_i} = v_i)$ and $s_i \in del(e_{a_j} = v_j)$, $\forall_{a_i,a_j}(e_{a_i} = v_i) \wedge (e_{a_j} = v_j) \rightarrow \neg$-$s_i$.

- Similary only a single action can be selected, $\forall_j \vee a_j$, $\forall_{j \neq k} a_j \rightarrow \neg a_k$.

- Finally, we encode the frame axioms with auxiliary variables with the state variable at the next time stage $s_i^{'}$, $\neg$+$s_i \wedge$ -$s_i \rightarrow (s_i \wedge s_i^{'}) \vee (\neg s_i \wedge \neg s_i^{'})$.

In summary, the extended SAT encoding introduces multi-valued effect variables to encode multiple outcomes for a probabilistic effects, and two auxiliary variables to encode the state transition resulting from the add effect and delete effect of some action. We used equality predicates for multi-valued effect variables to express a PPDDL instance with CNF clauses. The frame axioms, the mutual exclusive action constraint is the same as that of linear encoding of a PDDL instance. Translation of the initial belief state and multiple goal state will be discussed in the following section.



a	
0	1
1	1

a	φ	e	
0	0	no-op	1
0	0	v_1	0
0	0	v_2	0
0	1	no-op	1
0	1	v_1	0
0	1	v_2	0
1	0	no-op	1
1	0	v_1	0
1	0	v_2	0
1	1	no-op	0
1	1	v_1	P_1
1	1	v_2	P_2

Figure 2.4: Example of a CPT Converted From a Flat Probabilistic Effect. The nodes x, y, z are state variables, the node a is an action variable, ϕ is a CNF clause defined over three state variables, the node e is the effect variable that has two probabilistic outcomes with two conjunction of simple effects v_1 and v_2 with $p_1 + p_2 = 1$. The CPT for the action variable and the effect varialbes are presented next to the graph. The values for the action variables are normalized to 1 because the probabilistic query is conditioned on action variables. If an action was not chosen or the precondition was false, the effect must be no-op. Otherwise, the outcomes v_1 and v_2 are quantified by probability values.

2.3.2 Converting CNF clauses into Mixed DBN

The SAT encoding proposed in the previous section separates deterministic constraints explicitly from the probabilistic transitions. As the third step of compilation process in Figure 2.3, we present the conversion from CNF clauses into a mixed 2TDBN. The mixed 2TDBN $G =< V, F >$ is a directed graph with a set of variables V and a set of factor tables F that are either a conditional probability table or deterministic factor table. The set of variables V is simply the union of all variables introduced by SAT encoding, $V = S \cup A \cup \{+/\text{-}s_i\} \cup \{e_{a_i}\}$.

First of all, we identify the conditional probability tables in a PPDDL instance. The probabilistic information is only represented by the probabilistic effects. Assuming that all the

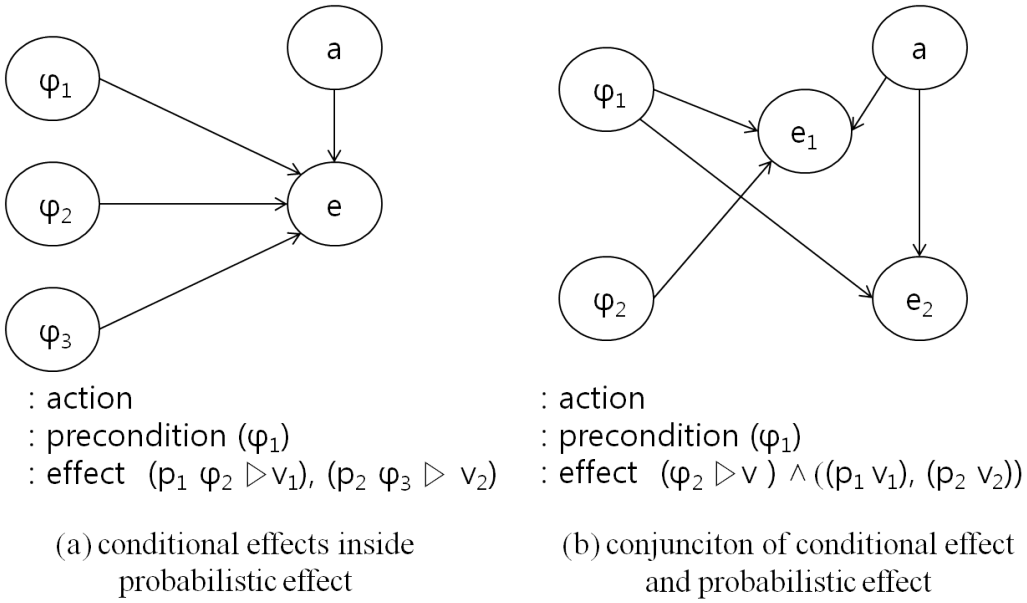


Figure 2.5: Example of CPT Converted From Nested Probabilistic Effect. (a) a probabilistic effect having conditional effects as outcomes, where ϕ_1 is the precondition variable that is 1 if the action precondition was satisfied, ϕ_2 and ϕ_3 are additional node for the conditions for the conditional effects, (b) conjunction of a conditional effect and a probabilistic effect, where two effect nodes were introduce to compile conjunction of two effects.

effects are expressed in the unary nondeterminism normal form, the possible outcome of an effect is no-op or conjunction of deterministic conditional effects that are annotated by probability values. In practice, it is beneficial to directly embed the tree structure of a nested effect inside the mixed 2TDBN with additional effect variables , which is shown in Figure 2.5. For simplicity, we assume that an action $a_i = (\phi, e_{a_i})$ has a precondition ϕ and a list of conjunction of simple effects e_{a_i} with n outcomes. In figure 2.4, a small example illustrates compiling probabilistic transitions from CNF clause, $a_i \wedge \phi_i \Leftrightarrow (e_{a_i} \neq \text{no-op})$. Clearly, the deterministic factor table enumerates boolean assignments to the state variables and the value of the ϕ is 1 if the assignment satisfies the CNF clause. Introducing the additonal node for ϕ is a choice of modeling, and the effect node could have the state variables as its parents. Two more examples for compiling probabilistic effect are given in figure 2.5. The first example shows the case where each outcome of the probabilsitic effect is deterministic conditional effect. Therefore, two additional nodes for condition

expressions are introduced and connected as parent nodes of the effect node. The second example is another case where the effect of an action is conjunction of a conditional effect and a probabilistic effect. The effect was not transformed to the unary nondeterminism normal form, but additional effect node is introduced to compile each effect separately. The conditional probability tables for both examples are not shown here but the idea is the same.

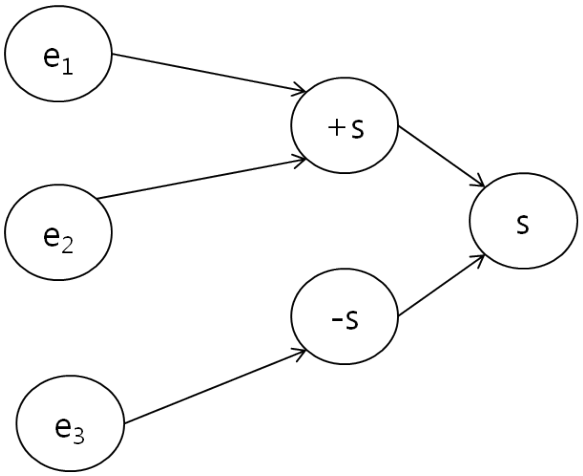
The next task is to compile deterministic constraints that are enforced implicitly by the PPDDL syntax. We can divide such deterministic constraints into three groups:

- 1. the state transitions clauses resulting from each outcome of an effect:
  (c1). for all e_{a_i} s.t. $\exists v \in Dom(e_{a_i})$ s.t. $s_i \in add(e_{a_i} = v)$, $\vee_{e_{a_i}}(e_{a_i} = v) \Leftrightarrow$ +s_i,
  (c2). for all $(e_{a_i} = v)$, $(e_{a_j} = w)$ s.t. $s_i \in add(e_{a_i} = v)$, $s_i \in add(e_{a_j} = w)$, $(e_{a_i} = v) \wedge (e_{a_j} = w) \rightarrow \neg$+$s_i$,
  (c3). for all e_{a_i} s.t. $\exists v \in Dom(e_{a_i})$ s.t. $s_i \in del(e_{a_i} = v)$, $\vee_{e_{a_i}}(e_{a_i} = v) \Leftrightarrow$ -s_i, and
  (c4). for all $(e_{a_i} = v)$, $(e_{a_j} = w)$ s.t. $s_i \in del(e_{a_i} = v)$, $s_i \in del(e_{a_j} = w)$, $(e_{a_i} = v) \wedge (e_{a_j} = w) \rightarrow \neg$-$s_i$

- 2. the frame axioms for each state variable,
  $\neg$+$s_i \wedge$ -$s_i \rightarrow (s_i \wedge s_i^{'}) \vee (\neg s_i \wedge \neg s_i^{'})$,

- 3. the mutual exclusivity constraint on action variables,
  $\forall_j \vee a_j, \forall_{j \neq k} a_j \rightarrow \neg a_k$,

The deterministic factor tables can be expressed in tabular format from the CNF clauses. They could have been stored in a compact CNF representation format like *DIMACS format* if the *UAI format* had allowed it. The auxiliary network for each type of the deterministic factors are illustrated by the examples in Figures 2.6-2.7. Figure 2.6 shows the auxiliary network for the state transition clauses. An effect node e is parent node of the +s node if



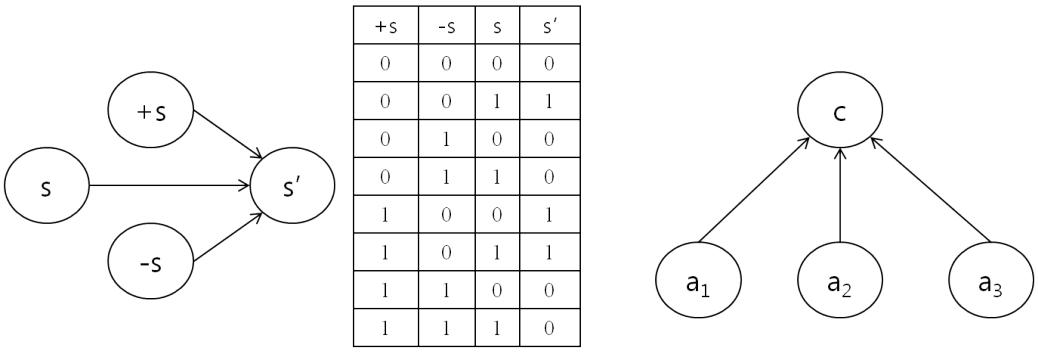
e_1	e_2	+s	
no-op	no-op	0	1
no-op	no-op	1	0
no-op	(s ∧ y)	0	0
no-op	(s ∧ y)	1	1
(s ∧ x)	no-op	0	0
(s ∧ x)	no-op	1	1
(s ∧ x)	(s ∧ y)	0	0
(s ∧ x)	(s ∧ y)	1	0

Figure 2.6: Auxiliary Network for the State Transition Clauses. Three effect nodes, e_1, e_2, and e_3, auxiliary variables, +s, -s, and state variable node s is shown in the graph. For brevity, the effect e_1 and e_2 are deterministic effects, and each of them has a single outcome $s \wedge x$ and $s \wedge y$ respectively. Since the positive literal s is shown in one of the outcomes of the effects e_1 and e_2, both nodes are connected to the auxiliary variable +s. The deterministic factor table for the node +s is shown next to the graph.

one of the outcomes of the effect variables contain positive literal of the state variable s. The parents of the -s node can be identified in similar way. As shown in the Figure 2.6, deterministic factor table can be obtained by simply rewriting each CNF clause in tabular format. Figure 2.7 illustrates the auxiliary network for the rest of the two constraints. The frame axioms are defined for each state variables, and the deterministic factor table is also tabular representation of the clause. The mutual exclusivity action constraint can be defined by introducing auxiliary variable c. As a whole, a PPDDL instance can be compiled into mixed 2TDBN.

In practice, auxiliary networks that are compiled from the state transition constraints or the mutual exclusivity axiom introduces large scope sized factor tables. For example, if there were m action variables, the factor table for the constraint has 2^{m+1} rows. In such cases, additional hidden variables are used to bound the maximum number of parent nodes as shown in Figure 2.8.

The final compilation task is to replicate the mixed 2TDBN up to desired time horizon, and



+s	-s	s	s'
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(a) Auxiliary network for the frame axiaom

(b) Auxiliary network for the mutual exclusivity constraint

Figure 2.7: Auxiliary Network for the Frame Axioms and the Mutual Exclusivity Axiom. (a) the frame axiom. The deterministic factor table is presented next to the graph. If two auxiliary variables were false, then the value of the future state variable must be the same as current value of the state variable. If one of the add effect or delete effect was true, then the future state variable will be true or false. Finally, the value of the last two rows are 0 since add effect and delete effect cannot occur simultaneously. (b) the mutual exclusivity constraint. The deterministic factor table for the node c is tabular representation of the mutual exclusivity constraint.

encode the initial belief state and the goal states. If the initial belief state was independently defined over the state variables, $Pr(s_0^0, s_1^0, \cdots, s_n^0) = Pr(s_0^0)Pr(s_1^0)\cdots Pr(s_n^0)$, each table for the state variable s_i^0 can be replaced with the initial probability distribution $Pr(s_i^0)$. On the other hand, if the initial belief state is defined jointly over the state variables, the joint distribution needs to be encoded by adding additional layer that relating the joint configuration of state variables. Similarly, the goal states can be quantified by adding additional layer that clipping the value of the inconsistent goal state to be 0. In case of having a single goal state, the additional layer can be absorbed into each table for state variables at the final stage.

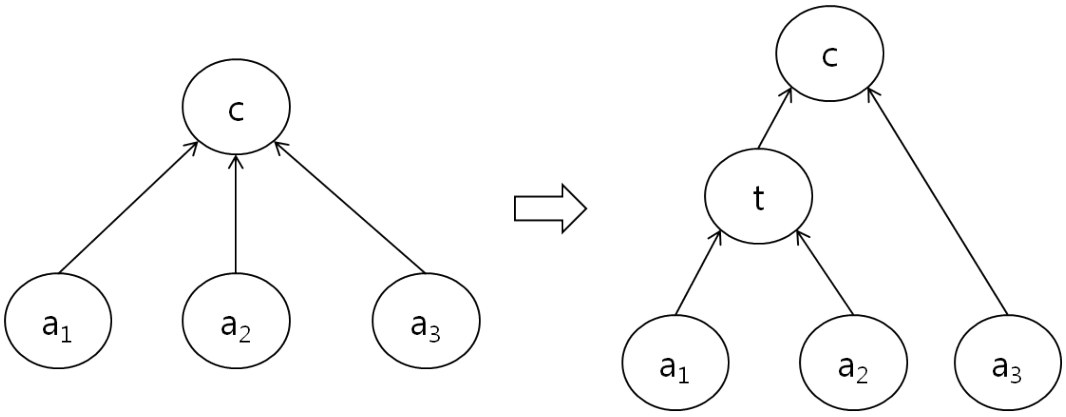


Figure 2.8: Bounding the Maximum Number of Parent Nodes. The hidden variable t was introduced to bound the maximum number of parent nodes.

2.3.3 Complexity of the Translation

In this section, the complexity of the tranlation is explained based on the input parameters that can be specified by the original PPDDL instance. Let $|AS|$ be the number of action schemata, $|PRE|$ be the number of predicates, p be the maximum number of parameters in the action schemata, q be the maximum number of variables in a predicate, and k be the number of ground object. The total number of ground state variables is $O(|PRE|k^q)$, and the total number of ground action variable is $O(|AS|k^p)$. Assuming that we have only flat effects, the number of effect variables are the same as the number of action variables. In addition, we introduce two auxiliary variables +s_i and -s_i for each state variable s_i. Thus, the total number of variables in a 2TDBN is $O(2|AS|k^p + 3|PRE|k^q)$. In case of introducing hidden variables to bound the maxmum in-degree of the mutual exclusivity constraint, we would introduce $|AS|k^p - 1$ hidden variables to bound the maximum in-degree of the constraint variable to be 2. Similarly, if the maximum number of parent nodes for the auxiliary variables was $|E|$, we would introduce $|E| - 1$ hidden variables. In such a case, the maximum arity of the CPTs is $\max(3, |\phi| + 1)$, where the $|\phi|$ is the maximum number of state variables shown in the precondition which contributes to the arity of the effect variable CPT. The maximum domain size is $v + 1$, where the v is the maximum

number of outcomes of a probabilistic effect. Finally, the total number of variables in the mixed 2TDBN is $O(3|AS|k^p + (1 + 2|E|)|PRE|k^q)$.

2.3.4 Compiling Slippery Gripper Problem

In this section, we demonstrate the compilation result of the slippery gripper problem [13] to provide a complete picture of the compliation process. Figure 2.9 is the slippery gripper problem expressed in PPDDL. Since the original PPDDL syntax does not allow the belief state, we added additional syntax for expressing the probability value for each state.

```
(define (domain ext-slippery-gripper)
  (:requirements :negative-preconditions :conditional-effects
                 :probabilistic-effects)
  (:predicates (gripper-dry) (holding-block) (block-painted)
               (gripper-clean))
  (:action pickup
       :effect (and (when (gripper-dry)
                          (probabilistic 0.95 (holding-block)))
                    (when (not (gripper-dry))
                          (probabilistic 0.5 (holding-block)))))
  (:action dry
       :effect (probabilistic 0.8 (gripper-dry)))
  (:action paint
       :effect (and (block-painted)
                    (when (not (holding-block))
                          (probabilistic 0.1 (not (gripper-clean))))
                    (when (holding-block)
                          (not (gripper-clean))))))
(define (problem ext-slippery-gripper)
  (:domain ext-slippery-gripper)
  (:init (gripper-clean)
         (probabilistic 0.7 (gripper-dry)))
  (:goal (and (gripper-clean) (holding-block) (block-painted))))
```

Figure 2.9: Slippery Gripper Problem Expressed in PPDDL. The PPDDL 1.0 cannot specify the initial belief state. Therefore, the syntax for specifying the initial belief is added to PPDDL.

The slippery gripper domain consists of four state variables, gripper-dry (gd), holding-

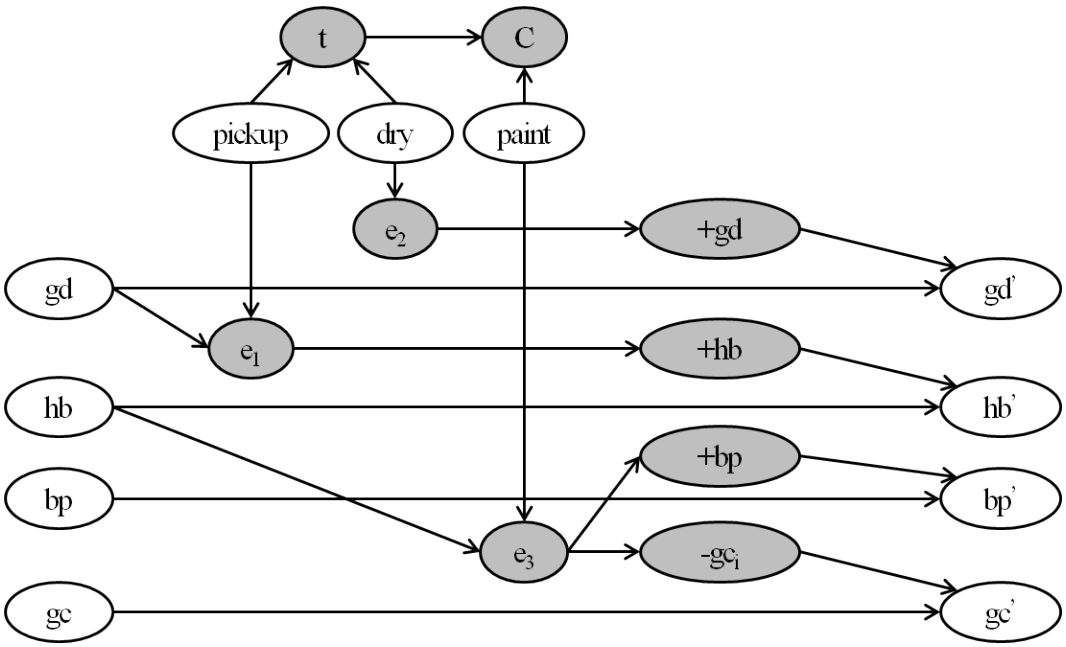


Figure 2.10: 2TDBN for the Slippery Gripper Problem. Shaded nodes indicates variables introduced by SAT encoding, and non-shaded nodes are directly mapped from ground predicates and ground actions.

block (hb), block-painted (bp), and gripper-clean (gc), and three actions, "pick up", "dry", and "paint". Initially, the gc is true, the gd is true with probability 0.7, and other two state variables are false. The goal is conjunction of lieterals, gd $\wedge$ hb $\wedge$ bp. If gripper-dry were true, the "pick up" action would add hold-block with probability 0.95. Otherwise hold-block would be added with probability 0.5. The "dry" action would add gripper-dry with probability 0.8. Lastly, If holding-block were true, gripper-clean will be deleted by "paint" action, and if it were false, gripper-clean would be deleted with probability 0.1. The "paint" action also adds block-painted. The compilation result of the slippery gripper problem into a mixed 2TDBN is presented in figure 2.10. The four state variables and three action variables are mapped to unshaded nodes in the mixed 2TDBN. The shaded nodes are introduced by the SAT encoding. Figure 2.11 presents all the conditional probability tables and deterministic tables in the mixed 2TDBN.

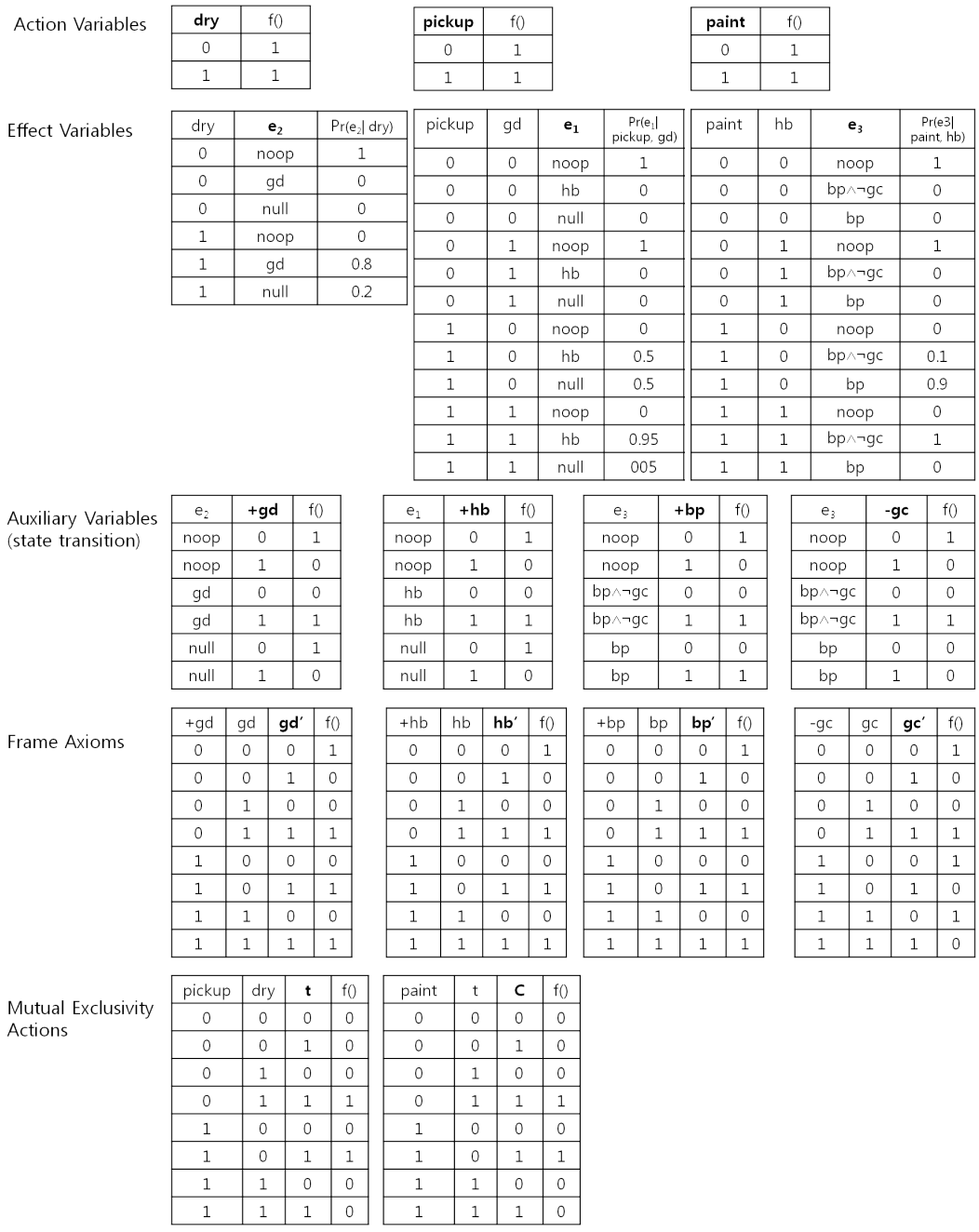


Figure 2.11: Slippery Gripper: Tables for its Mixed 2TDBN.

gd^0	
0	0.3
1	0.7

hb^0	
0	1
1	0

hb^L	
0	0
1	1

bp^0	
0	1
1	0

bp^L	
0	0
1	1

gc^0	
0	0
1	1

gc^L	
0	0
1	1

$+gd^L$	gd^{L-1}	gd^L	
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$+hb^L$	hb^{L-1}	hb^L	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$+bp^L$	bp^{L-1}	bp^L	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$-gc^L$	gc^{L-1}	gc^L	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Figure 2.12: Tables for the Initial Belief State and the Goal State. The superscript next to the name of the state variable denotes the time step. The probability distribution over the initial state variables, gd^0, hb^0, bp^0, gc^0, was factored into four separate tables, and the constraints for qualifying the goal state for each state variable at time step L is presented next to the table for each initial state variable. Since each goal constraint is a single variable function, the deterministic factor table at the L stage can aborb it as shown above.

The first three tables at the top of Figure 2.11 are conditional probability tables for the action variables, where the values were normalized to 1. It is possible to incorporate action prior. In such a case, the semantics of the planning problem and the probabilistic inference query must be properly addressed. The next three tables represent the probabilistic information in the slippery gripper domain. Note that if an action was not chosen, the effect of such choice would lead to *no-op*. The *null* effect of an action was introduced by the semantics the probabilistic effect of PPDDL specification. The next four tables compiles the CNF clauses stating the state transition axioms. The value of each effect shows the corresponding simple effect for each outcome. Considering the deterministic factor table for the auxiliary variable $+gd$, each row can be read as follows. The first row corresponds to the CNF clause that the add effect of the state variable gd is false if and only if the outcome of "dry" action was *no-op*. The fourth row correspondsto the CNF cluase that the add effect of the state variable gd is true if and only if the outcome of "dry" action was gd. The rest of the tables can be interpreted in the same way. The next four tables are truely tabular representation of the CNF clauses for the frame axioms, and the last two tables are also the

mutual exclusivity constraint rewritten in tabular format.

To find a length L probabilistic conformant plan, the mixed 2TDBN in Figure 2.10 needs to be extended up to L time horizon by simply replicating the same structure. At the final step of the compilation process, the initial belief state and the goal state need to be compiled to the extended mixed DBN. The tables for initial belief state and the goal state for the slippery gripper problem is presented in Figure 2.12. To incorporate the probability distributtion over the inital state variable, we replace tables at the first time step with the tables shown in Figure 2.12. The goal state can be encoded by the tables shown in Figure 2.12.

Chapter 3

Empirical Evaluation

In this chaper, we present expreimental results. The benchmark problems were selected from the past international planning competitions: *slippery gripper*, *comm*, and *blocks world*. Since the marginal MAP solver we applied can only read the *UAI format*, the problem instances were translated from PPDDL format into the *UAI format* by the compilation process explained in chapter 2. The experiment can be divided into two parts. The first part compares three marginal MAP algorithms, and the second part compares AOBB-JG, one of the marginal MAP algorithms we used, and Probabilistic-FF.

3.1 Benchmark Setup

We obtained the PPDDL domain definition files and problem instance files from the IPC 2004, 2006 website. The *slippery gripper* problem has only single instance definition file, but the *comm* problem has 25 instance definition files, where we converted the first instances for the experiment. The blocks world domain definition file was modified from the original version. In addition, we created a new instance definition files that is different from

PPDDL Domain	Source	Instance	Init. State	State Transition	Goal
Slippery Gripper	IPC 04	sg	Probabilistic	Probabilistic	Single state
Comm	IPC 06	p01	Nondeterministic	Deterministic	Single state
Blocks World	IPC 06	bw224	Deterministic	Probabilistic	Single state

Table 3.1: Overview of Benchmark Sets. Three PPDDL domain files were used for experiment. All the three problems have a single goal state that is expressed by a conjunction of state variables.

the instances used at the IPC 2006 probabilistic track. Since we formulated the probabilistic conformant planning task from the state transition model with a fixed time horizon, the *UAI files* were produced by incrementally increasing the time horizons from the shortest time horizon to a desired length.

- Slippery Gripper: This problem contains the probabilistic initial states and probabilistic state transition. The optimal plan for the slippery gripper problem should satisfy two partial order relations: the "paint" action precedes the "pick up" action, and the "dry" action precedes the "pick up" action. One such valid plan is $(\text{paint}, \text{dry}^i, \text{pick-up}^j)$, where the $i \geq 0$ and the $j \geq 1$ denotes the number of "dry" and "pick up" actions executed in the plan, and $T = 1 + i + j$. The probability of success for the plans satisfying above partial order relations is $0.9(d_i p_j + (1 - d_i)q_j)$, where $d_0 = 0.7$, $d_i = d_{i-1} + 0.8(1 - d_{i-1})$, $p_1 = 0.95$, $p_j = p_{j-1} + 0.95(1 - p_{j-1})$, $q_1 = 0.5$, and $q_j = q_{j-1} + 0.5(1 - q_{j-1})$. For example, the optimal length 3 plan is $(\text{paint, pick up, pick up})$ with the probability 0.830925. Thus, we can partly assure that the plans found by marginal MAP search algorithms were truely optimal by checking above equations on various i and j values.

- Comm: This class of problems is heavily deterministic, and it was used at the IPC 2006 conformant track. The state transition is deterministic and the uncertainity in the initial state is quantified by *one-of* clauses that we treated as a uniform distribution.

- Blocks World: The blocks world problem has many different versions, and we used the domain definition file released at the IPC 2006 probabilistic track. The original domain contains 7 action schemata, but we removed 3 action schemata. The removed action schemata is so called tower actions manipulating more than 1 blocks at once. The instance we used for the benchmark has 2 blocks.

3.2 Results from Marginal MAP Algorithms

In this section, we present the results from 3 marginal MAP algorithms developed recently [25, 15]: 1. AND/OR branch and bound search algorithm using weighted mini bucket heuristic with join graph cost shifting scheme (AOBB-JG) [6, 16, 14, 19, 15], 2. Branch and bound search algorithm using incremental mini cluster tree elimination heuristics (BBBTi) [15], and 3. Depth first branch and bound search algorithm using incremental joint tree upper bound with unconstrained variable orderings (Yuan) [25].

The algorithmic parameters are summarized as follows. The AOBB-JG has 3 parameters (i, c, j), where the i is the bound for mini bucket elimination, the c is the bound for AND/OR search space caching, and the j is the number of message passing iterations for join graph cost shifting scheme. We allowed 10 iterations for all experiments. In addition, AOBB-JG is equipped with a constraint propagation routines that exploits determinism inside the model by encoding the zero valued rows as a CNF clause and detect zero probability assignments by unit resolution. The BBBTi has 2 parameters the i bound and the c bound, and the Yuan's algorithm does not take any parameter. For AOBB-JG and BBBTi, we provided a topological variable ordering as a valid MAP ordering. The topological variable ordering was obtained as follows. The action variables are enumerated at the beginning of the ordering. Starting from the initial time stage, state variables, effect variables, hidden variables, and auxiliary variables are enumerated by following the directed edge between

nodes in the mixed DBN. All the algorithms were terminated after 6 hour (21,600 seconds) time limit and 4 GB memory limit. We tried various i-bound parameters for AOBB-JG and BBBTi, only the best result will be presented. The c-bound parameter was fixed to the same as the i-bound.

3.2.1 Slippery Gripper

Table 3.2 presents the problem statistics and results reported by each algorithm; L is the length of the plan, n is the total number of variables, m is the number of MAP variables, f is the number of factors, k is the maximum domain size, s is the maximum scope size, w^* is the induced width of the topological variable ordering, uw^* is the induced width of an unconstrained ordering found by minfill, h is the height of the pseudo tree from the constrained ordering, uh is the height from the unconstrained ordering, $sat\ vars$ is the number of SAT variables encoded by the constraint processing routine, $sat\ clauses$ is the number of SAT clauses, i.e., the number of zero valued rows. i-bd is the i-bound, c-bd is the cache bound , OR is the number of OR nodes explored by search, AND is the number of AND nodes explored, $pre\ time$ is the processing time, $total\ time$ is the total time, $Solution$ is the maximum probability of success, and $Bound$ is the initial upper bound.

The problem statistics shows that the size of the input problem is proportional to the length of the time horizon. Whereas the induced width of the unconstrained variable ordering remains around 6, the induced width of the constrained variable ordering increases with increase in the time horizon. Yuan's algorithm was the fastest overall and the BBBTi was at the second place. Considering the quality of the initial upper bound, AOBB-JG produced the best initial upper bound, and it solved problems up to 7 time horizon by inference.

Stats	L	n, m	f	k	s	w*, uw*	h, uh	sat vars	sat clauses
	2	42, 6	42	3	3	6, 4	13, 43	93	245
	3	61, 9	61	3	3	10, 5	18, 62	135	370
	4	80, 12	80	3	3	14, 6	22, 81	177	495
	5	99, 15	99	3	3	17, 6	27, 100	219	620
	6	118, 18	118	3	3	20, 6	30, 119	261	745
	7	137, 21	137	3	3	23, 6	34, 138	303	870
	8	156, 24	156	3	3	27, 6	38, 157	345	995
	9	175, 27	175	3	3	29, 7	43, 176	387	1120
	10	194, 30	194	3	3	32, 7	45, 195	429	1245

Algorithm	L	i-bd	c-bd	OR	AND	pre time	total time	Solution	Bound
AOBB-JG	2	28	28	0	0	0.01	0.01	0.7335	0.7335
	3	28	28	0	0	0.02	0.02	0.830925	0.830925
	4	28	28	0	0	0.14	0.14	0.884385	0.884385
	5	30	30	0	0	0.97	0.97	0.895077	0.895077
	6	28	28	0	0	8.33	8.33	0.898539	0.898539
	7	30	30	0	0	66.23	66.23	0.899618	0.899618
	8	20	20	14080	17119	3.38	4.16	0.899859	1.93198
	9	22	22	15188	19312	4.27	5.19	0.899967	1.43451
	10	28	28	29025	38468	46.94	49.29	0.899989	1.52619
BBBTi	2	28	28	47	49	0	0	0.7335	0.7335
	3	28	28	128	138	0	0.01	0.830925	2.26485
	4	28	28	119	127	0	0.01	0.884385	7.31411
	5	28	28	196	214	0.01	0.03	0.895077	11.6908
	6	28	28	330	367	0.02	0.08	0.898539	24.5902
	7	30	30	453	519	0.02	0.15	0.899618	71.3144
	8	28	28	405	451	0.02	0.29	0.899859	90.8221
	9	20	20	445	497	0.03	0.8	0.899967	138.556
	10	26	26	737	840	0.03	1.96	0.899989	371.314
Yuan	2	-	-	7	-	0	0	0.7335	0.7335
	3	-	-	12	-	0	0	0.830925	1.66162
	4	-	-	49	-	0.01	0.01	0.884385	7.60698
	5	-	-	146	-	0.01	0.01	0.895077	11.6908
	6	-	-	381	-	0.01	0.03	0.898539	34.2711
	7	-	-	1043	-	0.02	0.06	0.899618	71.55
	8	-	-	2210	-	0.02	0.11	0.899859	90.8221
	9	-	-	5190	-	0.03	0.26	0.899967	194.283
	10	-	-	15030	-	0.03	0.65	0.899989	521.865

Table 3.2: Empirical Evaluation with Sippery Gripper Problem. The number of state variables was 4, the number of action variables was 3, the total number of nodes was 23 in 2TDBN.

Stats	L	n, m	f	k	s	w*	h	sat var	sat clauses
	2	653, 94	653	2	5	103	140	1307	3671
	3	957, 141	957	2	5	155	198	1915	5826
	4	1261, 188	1261	2	5	207	270	2523	7981
	5	1565, 235	1565	2	5	259	324	3131	10136
	6	1869, 282	1869	2	5	311	375	3739	12291
	7	2173, 329	2173	2	5	363	436	4347	14446
	8	2477, 376	2477	2	5	415	488	4955	16601
	9	2781, 423	2781	2	5	467	540	5563	18756
Algorithm	L	i-bd	c-bd	OR	AND	pre time	total time	Solution	Bound
AOBB	2	2	2	0	0	1.18	1.18	0	0.00E+00
JG	3	4	4	0	0	3.07	3.07	0	0.00E+00
	4	4	4	0	0	7.13	7.13	0	4.50E+47
	5	6	6	0	0	11.09	11.09	0	0.00E+00
	6	2	2	2208	2234	17.8	19.06	0.25	1.09E+98
	7	2	2	79776	81574	25.83	74.81	0.25	3.15E+133
	8	2	2	2478	2480	34.98	36.96	0.25	2.13E+139
	9	2	2	6283	6641	49.04	54.87	0.25	1.86E+153

Table 3.3: Empirical Evaluation with Comm Problem. The number of state variables was 45, the number of action variables was 46, the total number of nodes was 349 in 2TDBN.

3.2.2 Comm

Table 3.3 presents the problem statistics and results reported by each algorithm. The AOBB-JG was the only algorithm that could solve the *comm* problem up to 9 time horizon. The induced width of the constrained ordering is 103 for the length 2 plan problem and 467 for the length 9 plan problem, which implies that heuristics generated by graph based algorithms may be uninformative. Furthermore, the only probabilistic tables in the problem are two state variables at the initial state, which means that AOBB-JG could solve the problem efficently by detecting the zero probability subplans early by constraint processing routines. The comm problem does not have solution up to 5 time horizon, thus AOBB-JG didn't search the problem because any plan leads to zero probability. The large induced width of the problem not only makes the heuristic inaccurate but also consumes huge amount of memory, so the i-bound was limited by 2 up to 9 time horizon and the solver was terminated due to memory from 10 time horizon.

3.2.3 Blocks World

Stats	L	n, m	f	k	s	w*, uw*	h, uh	sat var	clauses
BW224	3	201, 24	201	3	5	32, 17	54, 202	421	1719
	4	265, 32	265	3	5	40, 17	66, 266	555	2353
	5	329, 40	329	3	5	48, 17	78, 330	689	2987
	6	393, 48	393	3	5	57, 17	90, 394	823	3621
	7	457, 56	457	3	5	67, 17	99, 458	957	4255
	8	521, 64	521	3	5	73, 17	111, 522	1091	4889
	9	585, 72	585	3	5	85, 17	129, 586	1225	5523
	10	649, 80	649	3	5	88, 17	132, 650	1359	6157

algorithms	L	i	c	OR	AND	pre time	total time	Solution	Bound
AOBB JG	3	10	10	201	202	0.56	0.57	0.140625	1.410625
	4	10	10	2264	2294	0.9	1.06	0.5625	1.51E+09
	5	10	10	33601	34166	1.28	4.99	0.703125	1.16E+08
	6	12	12	441711	450030	3.62	66.91	0.808594	7.68E+16
	7	16	16	4767559	4872884	55.59	879.03	0.870117	1.45E+18
	8	18	18	46897433	48117132	224.61	9390.6	0.91626	3.04E+15
	9	10	10	80960476	81880618	2.57	out	nan	8.72E+19
	10	10	10	70629254	71552310	2.82	out	nan	1.09E+21
BBBTi	3	12	12	177	178	0.24	0.35	0.140625	5.13E+06
	4	12	12	846	875	0.38	1.95	0.28125	3.79E+10
	5	10	10	5181	5660	0.32	8.93	0.28125	1.46E+13
	6	12	12	80184	87724	0.64	242.19	0.808594	2.49E+17
	7	26	26	947040	1036077	1.86	18231.2	0.870117	1.83E+02
	9	22	22	4074	4169	29.95	out	0.943176	2.02E+03
	10	28	28	2024	2068	31.67	out	0.990327	1.80E+04
Yuan	3	-	-	25	-	5.51	7.53	0.140625	0.140625
	4	-	-	62	-	7.55	10.81	0.5625	1.47656
	5	-	-	1148	-	12.1	92.88	0.703125	8.96484
	6	-	-	11982	-	13.46	1029.82	0.808594	49.533
	7	-	-	209726	-	17.55	18809.1	0.870117	296.851
	8	-	-	247596	-	21.31	out	0.870117	702.582
	9	-	-	380441	-	23.08	out	0.885498	2691.55
	10	-	-	245637	-	27.55	out	0.931504	20239.9

Table 3.4: Empirical Evaluation with Blocks World Problem with 2 Blocks. If the algorithms was terminiated due to 6 hour time out, the best anytime solution is reported. The number of state variables was 9, the number of action variables was 8, the total number of nodes was 73 in 2TDBN.

Table 3.3 presents the problem statistics and results reported by each algorithm. The overall results show that AOBB-JG performed best, and BBBti was at the second place. Both AOBB-JG and BBBTi were able to find plans up to legth 8, and Yuan was out of time after

7 time horizon.

3.3 Comparion with Other Planners

In this section, we compare the marginal MAP search algorithms with COMPLAN and Probabilistic-FF. The performance of Probabilistic-FF was evaluated on the same benchmark envrionment, and we comapred the results with AOBB-JG.

To the best of our knowledge, COMPLAN is the most recent probabilistic conformant planning algorithm that maximizes the probability of success for a plan with a fixed time horizon. COMPLAN extends linear encoding of SATPLAN with chance variables that encode the uncertainity of a planning problem, and compiles CNFs into deterministic decomposable negation normal form [3], and then apply the marginal MAP query. The conformant plan is found by depth first branch and bound search guided by heuristics generated by a relaxed marginal MAP query on the d-DNNF compilation. Note that the induced width of constrained variable orderings for a planning problem is usually prohibited as shown in the previous section. AOBB-JG relaxes the problem based on the weighted mini-buckets with cost shifting, whereas COMPLAN allows an unconstrained variable ordering to yield lower induced width.

Probabilistic-FF finds a conformant plan that achieves the goal higher than a desired threshold by heuristic forward search in a belief state space. The belief states are a probability distribution over states reachable from initial belief state by applying a sequence of actions. Probabilistic-FF represents belief states by DBN proposed by [2], and the DBN is further compiled into weighted CNFs, to retrieve probabilitic queries via weigthed model counting [22]. In our compilation, all the action variables were nodes of the DBN such that the probabilistic inference directly query over the action variables after marginalizing the rest

of the variables. On the other hand, each DBN for the belief states of the Probabilistic-FF unrolls only small portion of the entire DBN that can be instantiated by applying a specific sequence of actions, and the weighted model counting computes only posterior probability so that the heuristic function evaluates the probability of achieving the goal.

slippery gripper						
pff (h1, w0)	θ	0.7335	0.830925	0.884385	0.895077	0.898539
	time	0.04	0.03	0.04	0.05	0.04
	length	3	4	5	6	8
	θ	0.899618	0.899859	0.899967	0.899989	0.899999
	time	0.05	0.04	0.07	0.11	out
	length	10	11	13	15	-
pff (h2, w1)	θ	0.7335	0.830925	0.884385	0.895077	0.898539
	time	0.03	0.19	0.42	1.22	1.27
	length	2	4	5	6	6
	θ	0.899618	0.899859	0.899967	0.899989	0.899999
	time	3.05	6.29	13.89	31.56	156.86
	length	7	8	9	10	12
AOBB JG	θ	0.7335	0.830925	0.884385	0.895077	0.898539
	time	0.01	0.02	0.13	0.98	8.33
	length	2	3	4	5	6
	θ	0.899618	0.899859	0.899967	0.899989	0.899999
	time	66.23	4.13	5.19	49.29	37.23
	length	7	8	9	10	12
blocks world - bw224						
pff (h1, w0)	θ	0.14065	0.5625	0.703125	0.808594	0.870117
	time	0.04	0.05	oom	oom	oom
	length	4	4	-	-	-
AOBB JG	θ	0.14065	0.5625	0.703125	0.808594	0.870117
	time	0.57	1.06	5	67	879
	length	3	4	5	6	7

Table 3.5: Comparison Between Probabilistic-FF and AOBB-JG. Algorithms were evaluated in two planning problmes, slippery gripper and blocks world with 2 blocks. The θ is the threshold value for the probability of success.

Table 3.5 compares Probabilistic-FF and AOBB-JG on the planning problems: *slippery gripper*, and *blocks world*. We set threshold values of the Probabilistic-FF from optimal conformant planning solutions for comparison. For the Probabilistic-FF we present the best result from 2 heuristic functions and 3 weight propagation schemes. For the detail about Probabilistic-FF planner, we refer to the Probabilistic-FF codes available at the author's

website. For the AOBB-JG, we present the total time and the plan length, and the problem statistics are the presented in Table 3.4.

- Slippery gripper: Probabilistic-FF solved all the instances around 0.1 seconds except the one with threshold 0.9. The length of the plan is typically longer than the optimal, but plans that are closer to the optimal could be found with increased time. For the slippery gripper problem, the threshold 0.9 is the maximum probability of success achievable. Considering the length of the plan, the performance of AOBB-WMB-JG is comparable to Probabilistic-FF in many problem instances. The WMB relaxation was exact, i.e., the i-bound was same as the induced width when the time horizon was less than 7. Thus, AOBB-JG was solved by inference.

- Blocks World: Probabilistic-FF was not able to find a plan longer than 4 due to the 4GB memory limit. As we discussed in previous sections, AOBB-WMB-JG ran out of the time when the time horizon grows higher than 8.

Chapter 4

Conclusions

In this thesis, we applied state of the art marginal MAP algorithms to solve probabilistic conformant planning problems. First, we formulated the task of finding the optimal probabilistic conformant plan as the marginal MAP probabilistic inference. Since probabilistic planning problems are expressed in PPDDL which is extended from de-facto standard PDDL for classical planning, we developed a translation that compiles a PPDDL instance into a mixed dynamic bayesian network. The compilation process can be divided into two parts. The first part encodes a PPDDL instance as CNF clauses that extract functional relationships in planning problems. Our SAT encoding is based on the linear encodings of PDDL with multi-valued effect variables that incorporate probabilistic outcomes of actions. The next part is to compile such CNF clauses into mixed DBN.

We selected three probabilistic planning problems to evaluate the performance of marginal MAP algorithms. The *slippery gripper*, *comm*, *blocks world* problem was taken from the international planning competition website. However, the *comm* problem is the only problem actually used in the planning competition. The marginal MAP algorithms applied to the empirical evaluation are AOBB-JG, BBBTi, and Yuan. The AOBB-JG algorithm refers

to AND/OR branch and bound search algorithm using weighted mini bucket heuristic with join graph cost shifting scheme, the BBBTi algorithm refers to branch and bound search algorithm using incremental mini cluster tree elimination heuristics, and the Yuan's algorithm refers to depth first branch and bound search algorithm using incremental joint tree upper bound with unconstrained variable orderings.

The empirical evaluation showed that the marginal MAP algorithms were able to find optimal probabilistic conformant plans given fixed time horizon. It is not conclusive to state what is the best algorithm based on current results since the algorithmic parameters such as the number of message passing iterations for AOBB-JG and the cache bound for AOBB-JG and BBBTi are not fully covered by the experiment. We also compared AOBB-JG and Probabilistic-FF. It is shown that Probabilistic-FF performed better than AOBB-JG for finding a plan that exceeds given threshold, but AOBB-JG was able to find the shortest plan faster than Probabilistic-FF.

It is worth mentioning several downsides of our compilation.

- Direct mapping from ground predicates and action schemata to binary variables gives large number of state and action variables. Therefore, the compilation is not scalable if the problem instance has many ground objects. In addition, the compiled graphs usually have by a large induced width not only due to the constrained variable ordering but also due to the large scope sized factors introduced by deterministic constraints. Clearly, we need compact compilation that scales better.

- The probabilsitic planning problems contains a huge number of deterministic constraints that could be exploited further. AOBB-JG was the only marginal MAP algorithm that sovled the *comm* problem. When the constraint processing did not employed, AOBB-JG was not able to solve any of the instances as other algorithms.

- The compiled network was stored as tables in the *UAI format* since it is the only

option for current marginal MAP algorithms. The size of the table is exponential in the scope size. Since the large scope sized constraints introduce such exponentially large tables, hidden variables were introduced to bound the maximum scope size. Nevertheless, the number of new hidden variables are also exponential in the number of ground objects.

We conclude this thesis with several proposals for future directions.

- Benchmark on harder problems: The three planning problems we addressed are easy problems; *slippery gripper* was already defined at the propositional level and it contains 4 state variables and 3 action variables. *Blocks world* were simpified by removing three action schemata in the orginal release to bound the number of ground variables at each time stage. The problem instances solved by marginal MAP algorithms were limited to contain at most three blocks and 7 time horizon. Therefore, our goal is to compile the planning domains in more compact way, and improve the probabilistic inference algorithms to exploit symmetry and determinism further. In addition, it is desired to evaluate the marginal MAP algorithms with a more comprehensive set of algorithmic parameters.

- Compact translation: More compact translation at the propositional level is desired to push the marginal MAP search algorithms to solve planning instances leaving a larger number of constant objects and longer time horizon at the ground level. The SAS+ based SAT encoding for the classical planning is known to produce more compact translation compared to the linear encoding of SATPLAN. Thus, the straightforward future work is to convert a STRIPS like PPDDL domain into finite domain representation (FDR) [9], and translate the new SAT encoding into the graphical model with proper extension.

- Compact representation: We were able to translate a PPDDL instance into the *UAI*

format. However, the potential of the mixed network was not fully exploited because all the contraints were stored as conditional probability tables. It is desired to extend current solvers to read input file formats other than the *UAI format*. This requirement will eventually call for the new input file format that subsumes existing formats for the constraint programming as well as decision diagrams.

- Need for lifting: Planning problems have two source of lifting. The time horizon which implies replicated structure over the time axis, and the state/action representation parameterized by obejct variables. It was shown that MDP/POMDP planning can be reformulated as a mixture model of unbounded number finite length DBNs [23] with flat state variables. Still, lifted inference for the parameterized random variables declared in both STRIPS like PPDDL and SAS+ based FDR has not gained much attention in the literature. We leave improving current algorithms in conjunction with lifted inference on parfactor graphs possibly translated from lifted planning probelm representations as future work.

- Extending the Probabilistic Queries to Other Planning Problems: First of all, the reward was ignored in our formulation. In practice, most planning problems are formulated with the probabilistic transition as well as reward function. Therefore it is desired to extend current probabilistic query to embrace the reward functions as MDP planning. In addition, other classes of probabilistic planning problems like partially observable probabilistic planning problems needs to be addressed.

Bibliography

[1] The 2008 UAI Probablistic Inference Evaluation. `http://graphmod.ics.uci.edu/uai08/`.

[2] C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 1:1–93, 1999.

[3] A. Darwiche. New advances in compiling CNF to decomposable negation normal form. In *Proc. of ECAI*, pages 328–332. Citeseer, 2004.

[4] R. Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–191, 2013.

[5] R. Dechter and D. Larkin. Hybrid processing of beliefs and constraints. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 112–119. Morgan Kaufmann Publishers Inc., 2001.

[6] R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153, 2003.

[7] C. Domshlak and J. Hoffmann. Probabilistic planning via heuristic forward search and weighted model counting. *J. Artif. Intell. Res.(JAIR)*, 30:565–620, 2007.

[8] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.

[9] M. Helmert. Concise finite domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5):503–535, 2009.

[10] J. Huang. Complan: A conformant probabilistic planner. *ICAPS 2006*, page 63, 2006.

[11] N. Hyafil and F. Bacchus. Utilizing structured representations and CSPs in conformant probabilistic planning. In *ECAI*, volume 16, page 1033, 2004.

[12] H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. *KR*, 96:374–384, 1996.

[13] N. Kushmerick, S. Hanks, and D. S. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1):239–286, 1995.

[14] R. Marinescu. *AND/OR Search Strategies for Combinatorial Optimization in Graphical Models*. PhD thesis, University of California, Irvine, 2008.

[15] R. Marinescu, R. Dechter, and A. Ihler. Improving marginal MAP search in graphical models. In *submitted to UAI*, 2014.

[16] R. Mateescu. *AND/OR Search Spaces for Graphical Models*. PhD thesis, University of California, Irvine, 2007.

[17] R. Mateescu and R. Dechter. Mixed deterministic and probabilistic networks. *Annals of mathematics and artificial intelligence*, 54(1-3):3–51, 2008.

[18] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL: the planning domain definition language. 1998.

[19] L. Otten. *Extending the Reach of AND/OR Search for Optimization in Graphical Models*. PhD thesis, University of California, Irvine, 2013.

[20] J. Rintanen. Expressive equivalence of formalisms for planning with sensing. In *ICAPS*, pages 185–194, 2003.

[21] S. Russell and P. Norvig. AI a Modern Approach. *Learning*, 2(3):4, 2005.

[22] T. Sang, P. Beame, and H. Kautz. Solving bayesian networks by weighted model counting. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, volume 1, pages 475–482, 2005.

[23] M. Toussaint. Probabilistic inference as a model of planned behavior. *Künstliche Intelligenz*, 3(9):23–29, 2009.

[24] H. L. Younes and M. L. Littman. PPDDL 1.0: The language for the probabilistic part of ipc-4. In *Proc. International Planning Competition*, 2004.

[25] C. Yuan and E. A. Hansen. Efficient computation of jointree bounds for systematic map search. In *IJCAI*, pages 1982–1989, 2009.