

# Faster Probabilistic Planning Through More Efficient Stochastic Satisfiability Problem Encodings

Stephen M. Majercik and Andrew P. Rusczeck

Department of Computer Science

Bowdoin College

Brunswick, ME 04011-8486

{smajerci, arusczeck}@bowdoin.edu

## Abstract

The propositional contingent planner ZANDER solves finite-horizon, partially observable, probabilistic planning problems at state-of-the-art-speeds by converting the planning problem to a stochastic satisfiability (SSAT) problem and solving that problem instead (Majercik 2000). ZANDER obtains these results using a relatively inefficient SSAT encoding of the problem (a linear action encoding with classical frame axioms). We describe and analyze three alternative SSAT encodings for probabilistic planning problems: a linear action encoding with simple explanatory frame axioms, a linear action encoding with complex explanatory frame axioms, and a parallel action encoding. Results on a suite of test problems indicate that linear action encodings with simple explanatory frame axioms and parallel action encodings show particular promise, improving ZANDER's efficiency by as much as three orders of magnitude.

## Introduction

Majercik (2000) showed that a compactly represented artificial intelligence planning domain can be efficiently represented as a *stochastic satisfiability* problem (Littman, Majercik, & Pitassi 2001), a type of Boolean satisfiability problem in which some of the variables have probabilities attached to them. This led to the development of ZANDER, an implemented framework that extends the planning-as-satisfiability paradigm to support contingent planning under uncertainty: uncertain initial conditions, probabilistic action effects, and uncertain state estimation (Majercik 2000).

There are different ways of encoding a probabilistic planning problem as an SSAT problem, however, and it is not obvious which encoding is best for which problem. In this paper, we begin to address the issue of producing maximally efficient SSAT encodings for probabilistic planning problems. In the next section, we describe ZANDER. In the next two sections, we describe four types of SSAT encodings of planning problems, analyze their size and potential benefits and drawbacks, and describe and analyze results using these encodings on a suite of test problems. In the final section, we discuss further work.

## ZANDER

In this section, we provide a brief overview of ZANDER. Details are available elsewhere (Majercik 2000). ZANDER works on partially observable probabilistic propositional planning domains consisting of a finite set  $P$  of distinct *state propositions* which may be `True` or `False` at any discrete time  $t$ . A set  $P' \subseteq P$  of state propositions is declared to be the set of observable propositions, and the members of  $O$ , the set of *observation propositions*, are the members of  $P'$  tagged as observations of the corresponding state propositions. Thus, each observation proposition has, as its basis, a state proposition that represents the actual state of the thing being observed.

A *state* is an assignment of truth values to the state propositions. An *initial state* is specified by an assignment to the state propositions. A probabilistic initial state is specified by attaching probabilities to some or all of the variables representing these propositions at time  $t = 0$ . *Goal states* are specified by a partial assignment to the set of state propositions; any state that extends this partial assignment is a goal state. Each of a finite set  $A$  of *actions* probabilistically transforms a state at time  $t$  into a state at time  $t + 1$  and so induces a probability distribution over the set of all states. The task is to find an action for each time  $t$  as a function of the value of observation propositions at times  $t' < t$  that maximizes the probability of reaching a goal state.

Previous versions of ZANDER used a propositional problem representation called the sequential-effects-tree representation (ST), which is a syntactic variant of two-time-slice Bayes nets (2TBNs) with conditional probability tables represented as trees (Majercik 2000). In the ST representation, each action  $a$  is represented by an ordered list of decision trees, the effect of  $a$  on each proposition represented as a separate decision tree. This ordering means that the tree for one proposition can refer to old *and* new values of previous propositions, thus allowing the effects of an action to be correlated. The leaves of a decision tree describe how the associated proposition changes as a function of the state and action, perhaps probabilistically.

We currently represent problems using the Probabilistic Planning Language (*PPL*). *PPL* is a high-level action language that extends the action language  $\mathcal{AR}$  (Giunchiglia, Kartha, & Lifschitz 1997) to support probabilistic domains. An ST representation can be easily translated into a *PPL*

representation (each path through each decision tree is replaced by a  $\mathcal{PPL}$  statement) but  $\mathcal{PPL}$  allows the user to express planning problems in a more natural, flexible, and compact format. More importantly,  $\mathcal{PPL}$  gives the user the opportunity (but does not require them) to easily express state invariants, equivalences, irreversible conditions, and action preconditions—information that can greatly decrease the time required to find a solution.

ZANDER converts the  $\mathcal{PPL}$  representation of the problem into a *stochastic satisfiability* (SSAT) problem. An SSAT problem is a satisfiability (SAT) problem, assumed to be in conjunctive normal form, with two types of Boolean variables—termed *choice* variables and *chance* variables (Majercik 2000)—and an ordering specified for the variables. A choice variable is like a variable in a regular SAT problem; its truth value can be set by the planning agent. Each chance variable, on the other hand, has an independent probability associated with it that specifies the probability that that variable will be `True`.

Choice variables can be thought of as being existentially quantified—we must pick a single, best value for such a variable—while chance variables can be thought of as “randomly” quantified—they introduce uncontrollable random variation which, in general, makes it more difficult to find a satisfying assignment. So, for example, an SSAT formula with the ordering  $\exists v \forall w \exists x \forall y \exists z$  asks for values of  $v$ ,  $x$  (as a function of  $w$ ), and  $z$  (as a function of  $w$  and  $y$ ) that maximize the probability of satisfaction given the independent probabilities associated with  $w$  and  $y$ . This dependence of choice variable values on the earlier chance variable values in the ordering allows ZANDER to map contingent planning problems to stochastic satisfiability. Essentially, ZANDER must find an assignment *tree* that specifies the optimal action choice-variable assignment given all possible settings of the observation variables (Majercik 2000).

The solver does a depth-first search of the tree of all possible truth assignments, constructing a solution subtree by calculating, for each variable node, the probability of a satisfying assignment given the partial assignment so far. For a choice variable, this is the maximum probability of its children. For a chance variable, the probability will be the probability weighted average of the success probabilities for that node’s children. The solver finds the optimal plan by determining the subtree that yields the highest probability at the root node.

ZANDER uses unit propagation (assigning a variable in a unit clause—a clause with a single literal—its forced value) and, to a much lesser extent, pure variable assignment (assigning the appropriate truth value to a choice variable that appears only positively or only negatively) to prune subtrees in this search. Also, although the order in which variables are considered is constrained by the SSAT-imposed variable ordering, where there is block of similar (choice or chance) variables with no imposed ordering, ZANDER considers those with the earliest time index first. This *time-ordered heuristic* takes advantage of the temporal structure of the clauses induced by the planning problem to produce more unit clauses. ZANDER also uses dynamically calculated success probability thresholds to prune branches of the

tree. We are currently working on incorporating learning to improve ZANDER’s performance.

## SSAT Encodings

The SSAT encoding currently used by ZANDER—a linear action encoding with classical frame axioms—and two of the alternate encodings described below—a linear action encoding with simple explanatory frame axioms and a parallel-action encoding—are similar to deterministic plan encodings described by Kautz, McAllester, & Selman (1996). A third encoding—a linear action encoding with complex explanatory frame axioms—contains elements of these two alternate encodings and arises due to the probabilistic actions in our domains.

In all cases, variables are created to record the status of actions and propositions in a  $T$ -step plan by taking three cross products: actions and times 1 through  $T$ , propositions and times 0 through  $T$ , and random propositions and times 1 through  $T$ . Let  $A$ ,  $P$ ,  $O$ , and  $R$  be the sets of actions, state propositions, observation propositions, and random propositions, respectively, and let  $\mathcal{A} = |A|$ ,  $\mathcal{P} = |P|$ ,  $\mathcal{O} = |O|$ , and  $\mathcal{R} = |R|$ . Let  $V$  be the set of variables in the CNF formula. Then:

$$|V| = (\mathcal{A} + \mathcal{P} + \mathcal{O} + \mathcal{R})T + \mathcal{P} \quad (1)$$

The variables generated by all but the random propositions are choice variables. Those generated by the random propositions are chance variables. Each variable indicates the status of an action, proposition, observation, or random proposition at a particular time. In the parallel-action encoding, two additional actions are produced for each proposition  $p \in P$  at each time: a *maintain- $p$ -positively* action and a *maintain- $p$ -negatively* action, which increases the number of variables in this encoding by  $2\mathcal{P}$ .

Conceptually, we need clauses that enforce initial/goal conditions and clauses that model actions and their effects. The second group divides into two subgroups: clauses that enforce (or not) a linear action encoding, and clauses that model the impact of actions on propositions. Finally, in this last subgroup, we need clauses that model the effects of actions both when they *change* the value of a proposition and when they leave the value of a proposition *unchanged* (the frame problem). In the following sections, we will describe the clauses in each encoding that fulfill these functions.

### Linear Action Encoding With Classical Frame Axioms

**Initial and Goal Conditions:** Let  $\text{IC}^+ \subseteq P$  ( $\text{IC}^- \subseteq P$ ) be the set of propositions that are `True/False` in the initial state, and  $\text{GC}^+ \subseteq P$  ( $\text{GC}^- \subseteq P$ ) be the set of propositions that are `True/False` in the goal state, where  $\text{IC}^+ \cap \text{IC}^- = \emptyset$  and  $\text{GC}^+ \cap \text{GC}^- = \emptyset$ . This generates  $\mathcal{O}(\mathcal{P})$  unit clauses:

$$\bigwedge_{p \in \text{IC}^+} (p^0) \wedge \bigwedge_{p \in \text{IC}^-} \overline{(p^0)} \wedge \bigwedge_{p \in \text{GC}^+} (p^T) \wedge \bigwedge_{p \in \text{GC}^-} \overline{(p^T)} \quad (2)$$

where superscripts indicate times.

**Mutual Exclusivity of Actions:** Special clauses marked as “exactly-one-of” clauses specify that exactly one of the

literals in the clause be `True` and provide an efficient way of encoding mutual exclusivity of actions. A straightforward propositional encoding of mutual exclusivity of  $n$  actions would require, for each time  $t$ , an action disjunction clause stating that one of the actions must be `True`, and  $\binom{n}{2} = \mathcal{O}(n^2)$  clauses stating that for each possible pair of actions, one of the actions must be `False`. In subsequent solution efforts, the assignment of `True` to any action would force the assignment of `False` to all the other actions at that time step, but at a cost of discovering and propagating the effect of the  $\mathcal{O}(n)$  resulting unit clauses. Depending on the implementation of the SSAT solver, the number of mutual exclusivity clauses could also slow the discovery of unit clauses. By tagging the action disjunction clause as an exactly-one-of-clause, we reduce the total number of clauses, and the solver can make the appropriate truth assignments to all actions in the clause as soon as one of them is assigned `True`. This generates  $\mathcal{O}(T)$  exactly-one-of clauses:

$$\bigwedge_{t=1}^T \left( \bigvee_{a \in A} a^t \right) \quad (3)$$

**Effects of Actions:** Describing the effects of actions on propositions generates one or two clauses for each  $\mathcal{PPL}$  action effect statement. If the statement is deterministic (a probability of 0.0 or 1.0), the statement generates a single clause modeling the action’s deterministic impact on the proposition given the circumstances described by the statement. For example, the following clause states that an error condition is created if a painted part is painted again:

$$\text{paint causes error withp 1.0 if painted} \quad (4)$$

The time indices are implicit; if the `paint` action is executed at time  $t$  and `painted` is `True` at time  $t - 1$ , then `error` will be `True` at time  $t$ . Each  $\mathcal{PPL}$  statement of the type:

$$a \text{ causes } p \text{ withp } \pi \text{ if } c_1 \text{ and } c_2 \text{ and } \dots \text{ and } c_m \quad (5)$$

where  $a \in A$ ,  $p \in P \cup O$ ,  $c_i \in P$ ,  $1 \leq i \leq m$ , and  $\pi$  is 0.0 or 1.0 generates  $\mathcal{O}(T)$  clauses:

$$\bigwedge_{t=1}^T \left( \overline{a^t} \vee \bigvee_{q \in P_{a+p}} \overline{q^{t-1}} \vee \bigvee_{q \in P_{a-p}} q^{t-1} \vee \overline{p^t} \right) \quad (6)$$

if  $\pi = 0.0$ , where  $P_{a+p} \subseteq P$  is the set of  $c_i$ s that appear positively in (5) and  $P_{a-p} \subseteq P$  is the set of  $c_i$ s that appear negatively in that statement. If  $\pi = 1.0$ , the final literal in each clause is  $p^t$  rather than  $\overline{p^t}$ .

If statement (5) is probabilistic ( $0.0 < \pi < 1.0$ ), the statement generates two clauses modeling the action’s probabilistic impact on the proposition given the circumstances described by that statement. An example will clarify this process. Suppose we have the following action effect statement:

$$\text{paint causes o-painted withp 0.4 if new painted} \quad (7)$$

Here, the modifier “new” changes the implicit time index of `painted` to be the same as the time index of the action; in

other words, “new painted” refers to the value of `painted` after the `paint` action has been executed.

Since the probability in this action effect statement is strictly between 0.0 and 1.0, a chance variable  $\mathbf{cv}_1$  associated with this probability will be generated along with two clauses, one describing the impact of the action if the chance variable is `True` and one describing its impact if the chance variable is `False`. For example, this statement would generate the following two implications for time  $t = 1$ , where time indices,  $-t$ , are added to the variables:

$$\text{paint-1} \wedge \text{painted-1} \wedge \mathbf{cv}_{0.4-1} \rightarrow \mathbf{o-painted-1} \quad (8)$$

$$\text{paint-1} \wedge \text{painted-1} \wedge \overline{\mathbf{cv}_{0.4-1}} \rightarrow \overline{\mathbf{o-painted-1}} \quad (9)$$

where  $\mathbf{cv}_{0.4-1}$  is the chance variable associated with this action effect, and the subscript indicates its probability. Negating the antecedent and replacing the implication with a disjunction produces two clauses:

$$\overline{\text{paint-1}} \vee \overline{\text{painted-1}} \vee \overline{\mathbf{cv}_{0.4-1}} \vee \mathbf{o-painted-1} \quad (10)$$

$$\overline{\text{paint-1}} \vee \overline{\text{painted-1}} \vee \mathbf{cv}_{0.4-1} \vee \overline{\mathbf{o-painted-1}} \quad (11)$$

Thus, each  $\mathcal{PPL}$  statement of the type:

$$a \text{ causes } p \text{ withp } \pi \text{ if } c_1 \text{ and } c_2 \text{ and } \dots \text{ and } c_m \quad (12)$$

where  $a \in A$ ,  $p \in P \cup O$ ,  $c_i \in P$ , and  $0.0 < \pi < 1.0$  generates  $\mathcal{O}(T)$  clauses:

$$\bigwedge_{t=1}^T \left( \overline{a^t} \vee \bigvee_{q \in P_{a+p}} \overline{q^{t-1}} \vee \bigvee_{q \in P_{a-p}} q^{t-1} \vee \overline{\mathbf{cv}_\pi^t} \vee p^t \right) \quad (13)$$

where  $P_{a+p} \subseteq P$  is the set of  $c_i$ s that appear positively in (12) and  $P_{a-p} \subseteq P$  is the set of  $c_i$ s that appear negated in that statement, and  $\mathbf{cv}_\pi$  is the chance variable generated by (12). The clauses in (13) model the effect of the action when the chance variable is `True`. A similar set of clauses, but with  $\mathbf{cv}_\pi^t$  instead of  $\overline{\mathbf{cv}_\pi^t}$  model the effect when the chance variable is `False`.

The set of propositions that determine the effect of an action  $a$ ,  $\bigcup_{p \in P_a} (P_{a+p} \cup P_{a-p})$ , where  $P_a$  is the set of propositions affected by  $a$ , determine the arity  $C$  of that action. These propositions in the action effects statements can be either positive or negative, which leads to  $2^C$  possible combinations of  $C$  propositions. If every action affects every proposition and every observation, both positively and negatively, then the number of action effects clauses per time step is  $\mathcal{O}(\mathcal{A}(\mathcal{O} + \mathcal{P})2^C)$ , where  $C$  is the maximum arity among all the actions in the domain. Since the number of propositions is always greater than or equal to the number of observations, this bound becomes  $\mathcal{O}(\mathcal{AP}2^C)$ . With  $T$  time steps, an upper-bound on the number of actions effects clauses is  $\mathcal{O}(T\mathcal{AP}2^C)$ .

**Classical Frame Axioms:** The fact that, for example, action  $a_1$  has no impact on proposition  $p_1$  is modeled explicitly by generating two action effect clauses describing this lack of effect: one clause describing  $a_1$ ’s nonimpact when  $p_1$  is positive and one describing  $a_1$ ’s nonimpact when  $p_1$  is negative. For every proposition  $p \in P \cup O$  that an action  $a \in A$  has no effect on,  $\mathcal{O}(T)$  frame axiom clauses are generated:

$$\bigwedge_{t=1}^T \left( \overline{a^t} \vee \overline{p^{t-1}} \vee p^t \right) \wedge \bigwedge_{t=1}^T \left( \overline{a^t} \vee p^{t-1} \vee \overline{p^t} \right) \quad (14)$$













