Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Evaluating Temporal Plans in Incomplete Domains

Daniel Morwood and **Daniel Bryce**
dan.morwood@aggiemail.usu.edu, daniel.bryce@usu.edu
Utah State University

Abstract

Recent work on planning in incomplete domains focuses on constructing plans that succeed despite incomplete knowledge of action preconditions and effects. As planning models become more expressive, such as in temporal planning, the types of incompleteness may not only change, but plans become more challenging to evaluate. The primary difficulty to temporal plan evaluation is accounting for temporal constraints that may not be satisfied under all interpretations of the incomplete domain. In this work, we formulate incomplete temporal plan evaluation as a generalization of the temporal consistency problem, called partial temporal consistency. We present a knowledge compilation approach that is combined with symbolic constraint propagation and model counting algorithms for counting the number of incomplete domain model interpretations under which a plan is consistent. We present an evaluation that identifies the aspects of incomplete temporal plans most impact performance.

Introduction

Automated planning requires complete and correct domain models that are the result of often costly knowledge engineering (Simpson, Kitchen, and McCluskey 2007; Vaquero et al. 2007) or machine learning (Wu, Yang, and Jiang 2007). With insufficient human expertise or data, the planning domain model can be incomplete (Garland and Lesh 2002). Recent work (Weber and Bryce 2011; Nguyen, Kambhampati, and Do 2010) has shown that it is possible to synthesize plans that are robust to the incomplete model. Such robust plans are similar to conformant plans, where instead of incomplete knowledge of the world state, we have incomplete knowledge of the actions. Plan quality is defined by the number of interpretations of the incomplete domain under which the plan will succeed, which is most naturally posed as a propositional model counting problem (Gomes, Sabharwal, and Selman 2009). This work extends this formalism to PDDL 2.1 level 3 temporal plans (Fox and Long 2003).

Counting the domain model interpretations under which a temporal plan will succeed requires expressing the plan with conditional temporal constraints along with propositional constraints describing our knowledge of the domain. While prior work (Tsamardinos, Vidal, and Pollack 2003; Schwalb, Kask, and Dechter 1994) considers testing the consistency of such constraints, we wish to count the number of satisfiable assignments to the uncontrollable propositional variables (representing our incomplete knowledge), where the implied temporal constraints are consistent.

Tsamardinos, Vidal, and Pollack (2003) solve a decision form of this problem, where the propositional variables are uncontrollable and the constraints are either weakly or strongly consistent with respect to *all* propositional models. Alternatively, Schwalb, Kask, and Dechter (1994) solve the satisfiability version of this problem where the propositional variables are controllable. We bridge these approaches by assuming that the propositional variables are partitioned into controllable and uncontrollable sets. We also address a corresponding counting problem (which we call partial consistency), where we must count the number of propositional models where the constraints are either weakly or strongly consistent, while maximizing over the controllable variables. Our approach involves compiling the propositional and temporal constraints into algebraic decision diagrams (ADDs) (Bahar et al. 1993), composing the ADDs in a symbolic version of the Floyd-Warshall algorithm (Dechter, Meiri, and Pearl 1991), and then counting the propositional models that imply consistent temporal problems.

We conduct several experiments to evaluate our approach that vary the incompleteness of the planning domain and the structure of the plans from highly concurrent to sequential. We find that our approach scales well to evaluate large temporal plans in incomplete temporal planning domains and is most dramatically effected by the number of possible preconditions and delete effects (which cause plan threats, and disjunction in our knowledge representation).

Our presentation is organized as follows. We review conditional temporal problems and consistency in the next sections, and follow with our contribution to evaluating partial consistency. We then show how incomplete plan evaluation is formulated as partial consistency, and empirically evaluate the algorithms in several plan evaluation scenarios. We end with a discussion of related work and a conclusion.

Conditional Temporal Problems

The combination of propositional and temporal constraints has been studied under the names conditional temporal prob-

lem (CTP) (Tsamardinos, Vidal, and Pollack 2003), and conditional temporal networks (CTNs) (Schwalb, Kask, and Dechter 1994). Where the former is used to condition time points (nodes) on uncertain observations in conditional planning, the latter conditions constraints (edges) on action choices. In both cases, the conditions are represented as propositional sentences, and have an effect of implying an underlying temporal problem. Aside from the difference in how the implied temporal problems are conditioned (based on either nodes or edges), the main difference lies in what constitutes a solution to a CTP or a CTN. Solutions to CTPs ensure that all propositional models imply a consistent temporal problem because the propositions are assumed uncontrollable. Solutions to CTNs ensure that a single propositional model exists where the implied temporal constraints are consistent because the propositions are assumed controllable. In the following, we rely on the CTN representation (i.e., conditioned edges), but subscribe more closely to the CTP notion of a solution. We further generalize the notion of consistency to partial consistency, where we count the conditions (propositional models) under which an implied network is consistent because not all models of the uncontrollable variables imply consistent temporal problems.

Conditional Temporal Networks: We define a Conditional Temporal Network (CTN) as a tuple $\langle B, V, E, \psi \rangle$ where

- B is a set of propositions that is the union of:
  - B_{nc}: propositions that are not controllable.
  - B_c: propositions that are controllable.
- V is a set of real-valued temporal variables.
- E is a set of conditional constraints of the form

$$c \Rightarrow v_i \xrightarrow{[l,u]} v_j$$

  where $v_i, v_j \in V$, $l, u \in \mathbb{R}$, and c is a controllable proposition $p \in B_c$ or logical true $\top$ (meaning the constraint is not conditional). The constraint states that if c is true, then time point v_i must occur between l and u time units before time point v_j. We assume that for each constraint, there exists an equivalent (implicit) constraint $c \Rightarrow v_j \xrightarrow{[-u,-l]} v_i \in E$ to simplify our discussion below.
- $\psi \in 2^{2^B}$ is a propositional sentence expressed over B, with a set of models $M(\psi) \subseteq 2^B$. Each model $m \in M(\psi)$ is defined as a pair (m_{nc}, m_c), where $m_{nc} \subseteq B_{nc}$ and $m_c \subseteq B_c$. We also refer to the sets of partial models $M_c(\psi) = \{m_c | (m_{nc}, m_c) \in M(\psi)\}$ and $M_{nc}(\psi) = \{m_{nc} | (m_{nc}, m_c) \in M(\psi)\}$.

We denote by $E(m)$, the set of temporal constraints implied by a model $m \in M(\psi)$. A temporal constraint is implied by a model of m if its condition is $\top$ or it is p and $p \in m$. A CTN is consistent (Schwalb, Kask, and Dechter 1994) (Thrm. 2) iff there is a model $m \in M(\psi)$, such that the temporal constraints $E(m)$ that are implied by the model are consistent – the well-known simple temporal problem (STP) (Dechter, Meiri, and Pearl 1991). We denote by the 0-1 indicator function $\delta(E(m))$ whether the constraints are consistent. Tsamardinos, Vidal, and Pollack (2003) define a similar notion called scenario projections. A scenario projection is a set of temporal variables (and related constraints) with satisfiable conditions.

Schwalb, Kask, and Dechter (1994) need only find a model of ψ to show consistency because each proposition is assumed controllable (i.e., $B_{nc} = \{\}$); Tsamardinos, Vidal, and Pollack (2003) relax this assumption such that $B_{nc} \neq \{\}$ and corresponds to observations possibly made at execution time. When there exists uncontrollable propositions, it is possible to define weak and strong consistency in terms of the set of implied temporal problems. We extend these definitions to CTNs following the intuitions in CTPs.

Consistency: A CTN is *weakly consistent* iff for every uncontrollable partial model $m_{nc} \in M(\psi)$, there exists a controllable partial model m_c where the implied simple temporal problem is consistent; formally,

$$\forall m_{nc} \in M_{nc}(\psi) \exists (m_{nc}, m_c) \in M(\psi) \delta(E(m_{nc}, m_c)) = 1$$

A CTN is *strongly consistent* iff there exists a controllable partial model where, with all corresponding uncontrollable partial models, it implies a consistent STP; formally,

$$\exists m_c \in M_c(\psi) \, \forall (m_{nc}, m_c) \in M(\psi) \, \delta(E(m_{nc}, m_c)) = 1$$

Weak consistency assumes that the uncontrollable propositions are observed prior to execution. Strong consistency assumes that the uncontrollable propositions are not observed.

Partial Consistency: When a network is not weakly or strongly consistent (wrt. the universal quantifiers) it is useful to gauge the degree to which it is consistent. To accomplish this we define partial weak and strong consistency of a CTN.

The *weak partial consistency* of a CTN is the number of uncontrollable partial models where a corresponding controllable partial model is consistent, formally

$$\sum_{m_{nc} \in M_{nc}(\psi)} \max_{(m_{nc}, m_c) \in M(\psi)} \delta(E((m_{nc}, m_c)))$$

replacing the quantifiers by a maximum and summation.

The *strong partial consistency* of a CTN is the maximum over controllable partial models of the number of uncontrollable partial models that are consistent, formally

$$\max_{m_c \in M_c(\psi)} \sum_{(m_{nc}, m_c) \in M(\psi)} \delta(E((m_{nc}, m_c)))$$

Partial Consistency Algorithms

We view partial consistency as a propositional model counting problem where each model can only be counted if it implies a consistent STP. Model counting algorithms can be broadly grouped into search or knowledge compilation (Gomes, Sabharwal, and Selman 2009). As noted by Tsamardinos, Vidal, and Pollack (2003), many of the models can imply the same or similar temporal problem; in such cases, search based approaches can be inefficient because they may solve many related temporal problems disjunctively. We develop a knowledge compilation approach that exploits the overlapping constraints involved in solving similar temporal problems by using ADDs.

Our approach involves testing the consistency of each STP implied by a model of ψ. We test the consistency of

Algorithm 1: Floyd-Warshall(D, V)

```
for v_k ∈ V do
    for v_i ∈ V do
        for v_j ∈ V do
            D_ij = min(D_ij, D_ik + D_kj);
```

these STPs simultaneously by generalizing each scalar entry of the Floyd-Warshall (Cormen, Leiserson, and Rivest 1990) distance matrix to become an ADD. As the Floyd-Warshall algorithm requires only minimization and summation operations, both of which are defined for ADDs, the algorithm remains otherwise unchanged. The ADDs encode which models of ψ imply an upper bound on the distance between two time points and are defined over the set of CTN propositions B. We start by reviewing the Floyd-Warshall algorithm and its application to STPs.

Floyd-Warshall for STPs: Dechter, Meiri, and Pearl (1991) show that a simple temporal problem $\langle V, E\rangle$ (equivalent to a CTN with $B = \{\}$ and only unconditional constraints in E) is consistent through a special form of constraint propagation using the Floyd-Warshall all-pairs shortest path algorithm. The approach defines a distance graph that is represented by an explicit distance matrix D with entries

$$D_{ij} = \begin{cases} u & : v_i \xrightarrow{[l,u]} v_j \in E, i < j \\ -l & : v_i \xrightarrow{[l,u]} v_j \in E, i > j \\ 0 & : i = j \\ \infty & : \text{otherwise} \end{cases}$$

The Floyd-Warshall algorithm (Algorithm 1) computes the minimum distance between all pairs of time points, and the STP is guaranteed consistent if the graph does not contain a negative cycle (i.e., there does not exist a $D_{ii} < 0$). If consistent, the tightened constraint on two time points is $v_i \xrightarrow{[-D_{ji}, D_{ij}]} v_j$, and a feasible schedule is an assignment to the time points respecting the tightened constraints.

Floyd-Warshall for CTNs: In theory, it is possible to test the consistency of each STP implied by a model of ψ in a CTN by enumerating the models consistent with ψ and following the explicit approach above for STPs; however, this can be inefficient when the set of constraints in the CTN includes many unconditional constraints. We propose a symbolic generalization of the approach above that can exploit the common constraints among the implied STPs. The intuition is to replace each scalar entry of the distance matrix by an ADD. Each ADD encodes the distances between two time points for all possible implicit STPs implied by models of ψ. Upon completing the symbolic Floyd-Warshall algorithm, we count the number of consistent STPs by determining which STPs implied by models of ψ have negative cycles in the distance graph. All operations from Floyd-Warshall summations and minimizations, to partial consistency summations and maximizations are carried out symbolically as ADD operations.

We define a distance $c_{ij}(e)$ with respect to each constraint $e \in E$ and time points $v_i, v_j \in V$

$$c_{ij}(e) = \begin{cases} c \to u; \infty & : c \Rightarrow v_i \xrightarrow{[l,u]} v_j \in E, i < j \\ c \to -l; \infty & : c \Rightarrow v_i \xrightarrow{[l,u]} v_j \in E, i > j \end{cases}$$

where the if-then-else notation $c \to u; \infty$ is used to denote an ADD node representing that if c is satisfied, then u is the distance, else ∞ is the distance.

There may be multiple conditional constraints that relate two time points, and we denote the minimum among all of the constraints by the ADD c_{ij}, defined as

$$c_{ij} = \begin{cases} \min\left(\min_{e \in E} c_{ij}(e), \infty\right) & : i \neq j \\ 0 & : i = j \end{cases}$$

At this point, we could define the entries of the symbolic distance matrix as $D_{ij} = c_{ij}$ and run the Floyd-Warshall algorithm. However, this would correspond to testing the consistency of an implicit STP for each model $m \in 2^B$, many of which may not satisfy ψ (i.e., they are irrelevant). We can limit ourselves to testing the consistency of only those STPs implied by models $m \in M(\psi)$ by further modifying the distance matrix entries so that all models $m \notin M(\psi)$ imply STPs that are guaranteed to be inconsistent. We can make an STP inconsistent if we define the distance between all pairs of time points as $-\infty$. We accomplish this by representing ψ by an ADD that maps all models $m \in M(\psi)$ to ∞ and all models $m \notin M(\psi)$ to $-\infty$ and then taking the minimum of it with each c_{ij}. We define the entries in the distance matrix as

$$D_{ij} = \min(\psi_\infty, c_{ij})$$

where $\psi_\infty = ((\psi \cdot 2) - 1) \cdot \infty$.

Computing Partial Consistency: At the completion of the Floyd-Warshall algorithm, we can inspect the diagonal elements D_{ii} of the symbolic distance matrix to gather the consistent implicit STPs (those STPs that are defined by the paths in the ADDs that do not terminate in negative leaf nodes). Using the ADD threshold operator, we define an 0-1 ADD $D_{ii}^{\geq 0}$ for each D_{ii} where all leaf nodes with value greater or equal to 0 are mapped to 1, and all other leaf nodes are mapped to 0 (i.e., mapping all models with consistent STPs to 1). The STP implied by a model is inconsistent if there exists one i where $D_{ii}^{\geq 0}$ maps the model to 0. Therefore, if we take the minimum over i of entries $D_{ii}^{\geq 0}$, the resulting ADD we call C, will map a model to 0 if there exists at least one i making the STP inconsistent, formally

$$C = \min_i D_{ii}^{\geq 0}$$

We denote by $C(m)$ the constant ADD resulting from substituting $\top$ for each propositional variable $b \in m$, and $\bot$ for each $b \notin m$ into C. If $C(m) = 1$ the STP implied by m is consistent, and is inconsistent otherwise, as stated by the following theorem.

Theorem 1. *For all $m \in M(\psi)$, the implied explicit STP defined over the variables in V and constraints $E(m)$ is consistent (i.e., $\delta(E(m)) = 1$) iff the implicit STP is consistent, (i.e., $C(m) = 1$).*

Proof. (Sketch) If we assume that the explicit STP implied by $m \in M(\psi)$ is consistent then each diagonal entry of the explicit Floyd-Warshall distance matrix is equal to zero. If $C(m) = 0$, then at the completion of the Floyd-Warshall algorithm there is an i in the explicit matrix where $D_{ii}(m) < 0$. However, we assumed $m \in M(\psi)$ and the implied explicit STP is consistent, which is a contradiction.

If we assume that $C(m) = 1$, then $m \in M(\psi)$ and for all i in the symbolic distance matrix, $D_{ii}(m) = 0$. If the explicit STP implied by m is inconsistent, then a diagonal entry D_{ii} of the explicit distance matrix must be negative. A negative entry in D_{ii} of the explicit matrix implies that in the symbolic distance matrix $D_{ii}(m) < 0$, which is a contradiction. □

We compute the weak partial consistency by

$$\sum_{B_{nc}} \max_{B_c} C$$

and strong partial consistency by

$$\max_{B_c} \sum_{B_{nc}} C$$

By summing over the uncontrollable propositions and maximizing over the controllable propositions (where both operations are similar to Boolean projection), we attain a scalar indicating the respective weak or strong partial consistency.

Example: The following example illustrates the formulation of a symbolic distance matrix corresponding to a plan (described in the next section). We define a CTN $\langle (B_{nc}, B_c), V, E, \psi \rangle$ where for $\epsilon > 0$:

- $B_{nc} = \{add^e(a(s_1), p), del^e(a(s_2), p), pre^s(a(s_\infty), p)\}$
- $B_c = \{s_2^e \prec s_1^e, s_\infty^s \prec s_2^e\}$
- $V = \{s_0^e, s_1^s, s_1^e, s_2^s, s_2^e, s_\infty^s\}$
- $E = \{\ s_2^e \prec s_1^e \Rightarrow s_2^e \xrightarrow{[\epsilon,\infty]} s_1^e,$
  $s_\infty^s \prec s_2^e \Rightarrow s_\infty^s \xrightarrow{[\epsilon,\infty]} s_2^e,$
  $\top \Rightarrow s_0^e \xrightarrow{[\epsilon,\infty]} v, v \in \{s_1^s, s_1^e, s_2^s, s_2^e, s_\infty^s\}$
  $\top \Rightarrow v \xrightarrow{[\epsilon,\infty]} s_\infty^s, v \in \{s_0^e, s_1^s, s_1^e, s_2^s, s_2^e\}$
  $\top \Rightarrow s_1^s \xrightarrow{[dur(s_1),dur(s_1)]} s_1^e,$
  $\top \Rightarrow s_2^s \xrightarrow{[dur(s_2),dur(s_2)]} s_2^e\}$
- $\psi = \big(\neg pre^s(a(s_\infty), p) \vee add^e(a(s_1), p)\big) \wedge$
  $\big(\neg del^e(a(s_2), p) \vee s_2^e \prec s_1^e \vee s_\infty^s \prec s_2^e\big)$

This CTN formulation of a plan uses temporal variables V for the start and end times of durative actions, controllable propositions B_c for step orderings for threat resolution, uncontrollable propositions B_{nc} for incomplete action features. The temporal constraints E denote plan step orderings and the propositional constraints ψ denote cases consistent with the incomplete knowledge about the domain where the plan will succeed (i.e., the open conditions are satisfied and there are no threats).

The distance matrix is defined as follows:

$$\begin{pmatrix}
\psi_0 & \psi_\infty & \psi_\infty & \psi_\infty & \psi_\infty & \psi_\infty \\
\psi_{-\epsilon} & \psi_0 & \psi_{d(s_1)} & \psi_\infty & \psi_\infty & \psi_\infty \\
\psi_{-\epsilon} & \psi_{-d(s_1)} & \psi_0 & \psi_\infty & D_{s_1^e, s_2^e} & \psi_\infty \\
\psi_{-\epsilon} & \psi_\infty & \psi_\infty & \psi_0 & \psi_{d(s_2)} & \psi_\infty \\
\psi_{-\epsilon} & \psi_\infty & \psi_\infty & \psi_{-d(s_2)} & \psi_0 & D_{s_2^e, s_\infty^s} \\
\psi_{-\epsilon} & \psi_{-\epsilon} & \psi_{-\epsilon} & \psi_{-\epsilon} & \psi_\infty & \psi_0
\end{pmatrix}$$

where the ADDs are defined

$$\begin{aligned}
D_{s_1^e, s_2^s} &= (s_2^e \prec s_1^s) \rightarrow \psi_{-\epsilon}; \psi_\infty \\
D_{s_2^e, s_\infty^s} &= (s_\infty^s \prec s_2^e) \rightarrow \psi_{-\epsilon}; \psi_\infty \\
\psi_k &= \psi \rightarrow k; -\infty
\end{aligned}$$

for $k \in \{-\epsilon, \infty, 0, d(s_1), -d(s_1), d(s_2), -d(s_2)\}$. We overload the if-then-else notation in the ψ_k ADD above for the sake of brevity by replacing the proposition tested in the if condition by a propositional sentence ψ that must be satisfiable to satisfy the if condition.

At the completion of the Floyd-Warshall algorithm, the diagonal entries of the matrix are:

$$\begin{aligned}
D_{s_0^e, s_0^e} &= 0 \\
D_{ii} &= (\psi \wedge \neg s_\infty^s \prec s_2^e \wedge s_2^e \prec s_1^s) \rightarrow 0; -k \\
&\quad \text{for } i \in \{s_1^s, s_1^e, s_2^s, s_2^e, s_\infty^s\}
\end{aligned}$$

where k is a possibly different, strictly positive constant for each i and each model not satisfying $\psi \wedge \neg s_\infty^s \prec s_2^e \wedge s_2^e \prec s_1^s$. After mapping negative ADD leaves to zero and non-negative leaves to one, we have the threshold ADDs:

$$\begin{aligned}
D^{\geq 0}_{s_0^e, s_0^e} &= 1 \\
D^{\geq 0}_{ii} &= (\psi \wedge \neg s_\infty^s \prec s_2^e \wedge s_2^e \prec s_1^s) \rightarrow 1; 0 \\
&\quad \text{for } i \in \{s_1^s, s_1^e, s_2^s, s_2^e, s_\infty^s\}
\end{aligned}$$

Finally, taking the minimum among the threshold ADDs, we compute C:

$$\begin{aligned}
C &= \min(D^{\geq 0}_{s_0^e, s_0^e}, D^{\geq 0}_{s_1^s, s_1^s}, D^{\geq 0}_{s_1^e, s_1^e}, D^{\geq 0}_{s_2^s, s_2^s}, D^{\geq 0}_{s_2^e, s_2^e}, D^{\geq 0}_{s_\infty^s, s_\infty^s}) \\
&= (\psi \wedge \neg s_\infty^s \prec s_2^e \wedge s_2^e \prec s_1^s) \rightarrow 1; 0
\end{aligned}$$

Any model that satisfies $(\psi \wedge \neg s_\infty^s \prec s_2^e \wedge s_2^e \prec s_1^s)$ can be counted toward the partial consistency count. We compute the weak partial consistency by first maximizing C over the controllable variables B_c (and expanding ψ):

$$\max_{B_c} C = \big(\neg pre^s(a(s_\infty), p) \vee add^e(a(s_1), p)\big) \rightarrow 1; 0$$

Summing over B_{nc} computes the weak partial consistency:

$$\sum_{B_{nc}} (\neg pre^s(a(s_\infty), p) \vee add^e(a(s_1), p)) \rightarrow 1; 0 = 6$$

That is, there are six partial models m_{nc} in $M_{nc}(\neg pre^s(a(s_\infty), p) \vee add^e(a(s_1), p))$.

Computing strong partial consistency involves first summing over the uncontrollable variables:

$$\begin{aligned}
\sum_{B_{nc}} C &= \sum_{B_{nc}} (\psi \wedge \neg s_\infty^s \prec s_2^e \wedge s_2^e \prec s_1^s) \rightarrow 1; 0 \\
&= \neg s_\infty^s \prec s_2^e \wedge s_2^e \prec s_1^s \rightarrow 6; 0
\end{aligned}$$

Finally, maximizing over the controllable variables computes the strong partial consistency:

$$\max_{B_c} \neg s^s_\infty \prec s^e_2 \wedge s^e_2 \prec s^s_1 \rightarrow 6; 0 = 6$$

In this example, the weak and strong partial consistency are identical. This happens because there is only one assignment to the controllable variables that implies consistent STPs (of which there are six). In practice, when there are multiple satisfying assignments to the controllable variables, each will imply a potentially different number of consistent STPs, making weak and strong partial consistency differ.

Temporal Planning in Incomplete Domains

Planning in incomplete domains (Weber and Bryce 2011; Nguyen, Kambhampati, and Do 2010) involves actions with incompletely specified preconditions and effects, and plans are evaluated with respect to how many interpretations of the incomplete domain under which they succeed. While the problem has been studied by extending the classical planning model, its analog in the temporal planning model has been previously unstudied. In the following, we define the incomplete temporal planning model, and show how temporal plans are evaluated in the partial consistency framework.

Incomplete Temporal Domains: We extend the PDDL 2.1 (Fox and Long 2003) model of temporal actions to allow incomplete knowledge of the actions; our formalism largely builds upon incomplete classical planning domains (Weber and Bryce 2011). An incomplete temporal planning domain defines the tuple $\langle P, I, G, A, F, \phi \rangle$, where P is a set of state propositions, I is a set of initial state propositions, G is a set of goal propositions, A is a set of durative actions, F is a set of propositions (disjoint from P) describing the incomplete action features, and ϕ is a propositional sentence over F describing the incomplete knowledge about the actions. In this work, we assume that each action's duration $d(a)$ is known (leaving unknown durations for future work).

The set F consists of propositions of the form $pre^\gamma(a,p)$, $add^\tau(a,p)$, and $del^\tau(a,p)$, where $\gamma \in \{s,e,i\}$ and $\tau \in \{s,e\}$, which indicate the standard PDDL 2.1 relationships between a state proposition p and the time during the action a, where it is a precondition, add effect, or delete effect. The superscript denotes the start s, end e, or entire interval i of the action's duration. The models of the sentence ϕ are defined in terms of the propositions in F, and each model of ϕ corresponds to an interpretation of the incomplete domain $F^i \subseteq F$. Each interpretation is a normal PDDL 2.1 domain. We say that a feature $f \in F$ is unknown if $\phi \not\models f$ and $\phi \not\models \neg f$, and is otherwise known. In practice, we omit the known action features from F and ϕ and reason about them separately, but we find it more parsimonious to describe the action representation as a propositional sentence.

Temporal Plans: A plan π is a tuple $\langle \mathcal{S}, \mathcal{L}, \mathcal{O} \rangle$, where $\mathcal{S}$ is a set of steps, $\mathcal{L}$ is a set of causal links, and $\mathcal{O}$ is a set of temporal constraints on the start and end times of each step in $s \in \mathcal{S}$, which we denote by s^s and s^e respectively. A causal link $s_i \xrightarrow{p@[\gamma,\tau]} s_j$ states that p is a precondition of s_j at time $\tau \in \{s,e,i\}$ satisfied by an effect of s_i at time $\gamma \in \{s,e\}$. All plans contain dummy steps s_0 and s_∞ where the effects of s_0 are the initial state and the preconditions of s_∞ are the goals.

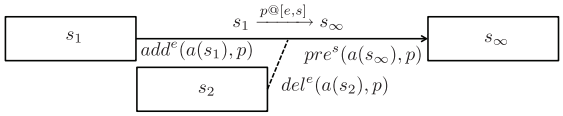

Figure 1: Example Plan

For example, the plan in Figure 1 consists of steps $\mathcal{S} = \{s_0, s_1, s_2, s_\infty\}$ (s_0 not shown), causal links $\mathcal{L} = \{s_1 \xrightarrow{p@[e,s]} s_\infty\}$, and temporal constraints in $\mathcal{O}$ that relate the start and end of each step and the end of step s_1 with the start of s_∞. The features are $F = \{add^e(a(s_1),p), del^e(a(s_2),p), pre^s(a(s_\infty),p)\}$, where $a(s)$ is the action applied in the step; the features indicate that $a(s_1)$ might add p at its end, $a(s_2)$ might delete p at its end, and s_∞ might require p as a precondition at its start. The knowledge $\phi = \top$ means that we have no information about whether the features in F are actual preconditions and effects or not.

Encoding Temporal Plans as CTNs: Encoding a temporal plan as a CTN allows us to compute the quality of the plan in terms of its weak or strong partial consistency. The set of temporal variables V includes variables for the start s^s and end s^e time of each step $s \in \mathcal{S}$. The set of propositions are:

$$B_{nc} = F$$
$$B_c = \{s^\mu_3 \prec s^\gamma_1, s^\tau_2 \prec s^\mu_3 | del^\mu(a(s_3),p) \in F, s_1 \xrightarrow{p@[\gamma,\tau]} s_2 \in \mathcal{L}\}$$

where each pair of propositions $s^\mu_3 \prec s^\gamma_1, s^\tau_2 \prec s^\mu_3$ in B_c indicate whether time point s^μ_3 (a threat) is promoted or demoted. In the example, $B_c = \{s^e_2 \prec s^e_1, s^s_\infty \prec s^e_2\}$.

The conditional temporal constraints E are defined as follows. Each temporal constraint $o \in \mathcal{O}$ is an unconditional constraint that states i) the duration of a step in terms of its start and end times, ii) orderings between time points corresponding to the producer and consumer of a causal link, and iii) promotion or demotion of a step known to threaten a causal link. There are also conditional constraints in E for every promotion/demotion variable in B_c of the form

$$s^\mu_1 \prec s^\tau_2 \Rightarrow s^\mu_1 \xrightarrow{[\epsilon,\infty]} s^\tau_2$$

where $\epsilon > 0$. The example from the previous section includes an exhaustive list of the conditional and unconditional constraints for the plan in Figure 1.

The propositional constraints ψ are defined in terms of whether the goal step s_∞ succeeds, which we denote by an indicator proposition $\eta(s_\infty)$ (and similarly for other steps). We use these indicator propositions to handle circular definitions that occur when two steps can possibly threaten each other's supporting causal links. We define a success axiom for each step, denoted $\psi(s_i)$. Each axiom defines the success of a step $\eta(s_i)$ as the step having each of its preconditions satisfied by a successful causal link, so that

$$\psi(s_i) := \eta(s_i) \Leftrightarrow \bigwedge_{\substack{p \in P, \\ \tau \in \{s,e,i\}}} pre^\tau(a(s_i),p) \Rightarrow \bigvee_{l \in \mathcal{L}(\pi)} \psi(l)$$

where $l = s_j \xrightarrow{p@[\gamma,\tau]} s_i$ and $\psi(l)$ represents the cases under which the causal link l succeeds (i.e., the step s_j succeeds and no step s_k clobbers the link). We note that if there are no causal links supporting the precondition, then the disjunction is equal to logical false, $\bot$. In the example, steps s_1 and s_2 have no preconditions, so $\psi(s_1) = \top$, $\psi(s_2) = \top$ and

$$\psi(s_\infty) := \eta(s_\infty) \Leftrightarrow pre^s(a(s_\infty), p) \Rightarrow \psi(l_1)$$

where $l_1 = s_1 \xrightarrow{p@[e,s]} s_\infty$. The success of a causal link $\psi(l)$ is defined as whether the producing step s_j succeeds and adds the effect, and the link is not threatened:

$$\psi(l) := \eta(s_j) \wedge add^\gamma(a(s_j), p) \wedge \bigwedge_{s_k \in \mathcal{S} \setminus \{s_0, s_\infty\}} nt(s_k, l)$$

where $nt(s_k, l)$ identifies the domain interpretations where s_k is not a threat to link l. In the example,

$$\psi(l_1) := \eta(s_1) \wedge add^e(a(s_1), p) \wedge nt(s_2, l_1)$$

A step is not a threat to a link $nt(s_k, l)$, as defined by

$$nt(s_k, l) := \neg\eta(s_k) \vee \neg del^\mu(a(s_k), p) \vee s_k^\mu \prec s_j^\gamma \vee s_i^{\tau''} \prec s_k^\mu$$

where the step s_k must fail, not delete p at time $\mu \in \{s, e\}$ between times s_j^γ and $s_i^{\tau''}$, and $\tau'' = e$ if $\tau \in \{e, i\}$ or $\tau'' = s$ if $\tau = s$. In the example,

$$nt(s_2, l_1) := \neg del^e(a(s_2), p) \vee s_2^e \prec s_1^e \vee s_\infty^s \prec s_2^e$$

We define ψ (in the CTN representing the plan) in terms of the success of s_∞ and our knowledge ϕ, but must ensure that we do not include the $\eta(s_i)$ variables in the partial consistency counts because ψ must be defined over B. To express ψ in terms of B, we substitute $\top$ for each instance of $\eta(s_\infty)$ because we want to count the models where the plan succeeds, and we existentially abstract the other $\eta(s_i)$ variables (i.e., project onto the variables in B), so that

$$\psi := \exists_{\{\eta(s_i) | s_i \in \mathcal{S}\}} \psi(s_\infty)[\top / \eta(s_\infty)] \wedge \phi$$

where existential abstraction is defined: $\exists_S f = \exists_{S \setminus s} \exists_s f$ and $\exists_s f = f[\top/s] \vee f[\bot/s]$.

From the example, after substitution and existential abstraction, we define

$$\begin{aligned} \psi &:= \exists_{\{\eta(s_i) | s_i \in \mathcal{S}\}} \psi(s_\infty)[\top / \eta(s_\infty)] \wedge \phi \\ &= \big(\neg pre^s(a(s_\infty), p) \vee add^e(a(s_1), p)\big) \wedge \\ &\quad \big(\neg del^e(a(s_2), p) \vee s_2^e \prec s_1^e \vee s_\infty^s \prec s_2^e\big) \end{aligned}$$

which states that either p is not a precondition of s_∞, or s_1 adds p, and at least one of the following: s_2 does not delete p, s_2 ends prior to s_1 ending, or s_2 ends after the start of s_∞ (which will not occur because the implied temporal constraints will be inconsistent).

We can count the number of interpretations of an incomplete domain under which a plan is valid by formulating the plan as a CTN and evaluating its partial consistency. The following theorem establishes that an interpretation of the incomplete domain and a particular choice of promotions/demotions is valid iff the corresponding model of the CTN implies a consistent STP.

Theorem 2. *Plan π has no flaws under the domain interpretation F^i iff in the CTN corresponding to π, when $m_{nc} = F^i$, there exists a partial model m_c such that $C((m_{nc}, m_c)) = 1$.*

Proof. (Sketch) If we assume that π has no flaws under the domain interpretation F^i, then there are no open conditions or threatened links. If $m_{nc} = F^i$ and there does not exist an $m_c \in M_c(\psi)$ where $C((m_{nc}, m_c)) = 1$, then by Theorem 1, it must be the case that $(m_{nc}, m_c) \notin M(\psi)$ (if there is an open condition) or the STP implied by (m_{nc}, m_c) is inconsistent (if there is a threat), which is a contradiction.

Assume that $m_{nc} = F^i$ and there exists a partial model m_c such that $C((m_{nc}, m_c)) = 1$. If π has a flaw under the domain interpretation F^i, then there is an open condition or threatened link. If there is an open condition, then $(m_{nc}, m_c) \notin M(\psi)$ and $C((m_{nc}, m_c)) = 0$, and if there is a threat, then there is not a m_c such that $C((m_{nc}, m_c)) = 1$, which is a contradiction. □

It follows from Theorem 2 that the number of domain interpretations under which a plan is valid is equivalent to the weak or strong partial consistency of the CTN formulated from the plan.

Evaluation

We evaluate our approach by using it to assess the quality of the plans generated by the fastest temporal planners for each instance solved during the 2011 International Planning Competition (IPC). We augment each instance by injecting incomplete domain features that affect the plan, and then compute partial consistency. In the following, we discuss the test cases chosen for the evaluation, implementation details, the evaluation metrics, and the empirical results.

Test Cases: In our first experiment we select from the 2011 IPC instances a representative small, medium, and large plan from three domains: Matchcellar (instances 000, 017, and 013), TMS (instances 000, 001, and 014), and Sokoban (instances 000, 005, and 004). These domains represent alternative plan structures that provide some insight to our evaluation. Plans in the Matchcellar domain admit sequences of tightly constrained concurrent actions (i.e., walking and changing a fuse while holding a match for each possible fuse). TMS plans involve multiple concurrent, yet relatively unconstrained actions. Sokoban involves sub-plans for multiple agents that are sequential, but have multiple constraints on how agents can interact concurrently. Matchcellar is relatively sequential, Temporal Machine Shop is mostly concurrent, and Sokoban has aspects of both.

We attain incomplete domains by randomly introducing (over ten random seeds) unknown features that impact the plans. To introduce a possible add effect, we uniformly sample a causal link $s_i \xrightarrow{p@[\gamma,\tau]} s_j$ from the plan and add the proposition $add^\gamma(a(s_i), p)$ to F (and similarly for preconditions). For delete effects, we uniformly sample a causal link $s_i \xrightarrow{p@[\gamma,\tau]} s_j$ and an action a from the plan, and add $del^e(a, p)$ to F. In each instance we add incomplete features in multiples of three because we add a possible precondition, add effect, and delete effect together.

	$\|F\|$	$E[F(l)]$	$T(\psi_\infty)$	$T(FW)$	$T(SC)$	$T(WC)$
m_1	21	0.81	0.01	0.00	0.00	0.00
m_1	42	1.70	4.15	0.17	1.36	4.79
m_1	63	2.51	OOT	OOT	OOT	OOT
m_2	21	0.36	0.01	0.01	0.00	0.00
m_2	42	0.83	7.74	4.41	4.77	6.97
m_2	63	1.20	OOT	OOT	OOT	OOT
m_3	21	0.22	0.04	0.15	0.00	0.00
m_3	42	0.50	3.46	28.79	0.21	1.20
m_3	63	0.57	298.58	7.45	15.76	55.63
s_1	21	0.13	0.00	0.00	0.00	0.00
s_1	42	0.28	6.38	0.03	0.32	0.13
s_1	63	0.40	57.26	0.92	17.09	8.77
s_2	21	0.10	0.02	0.00	0.00	0.00
s_2	42	0.21	0.81	0.01	0.05	0.02
s_2	63	0.31	122.06	0.51	11.23	5.66
s_3	21	0.05	0.24	0.01	0.00	0.00
s_3	42	0.09	1.41	0.02	0.03	0.03
s_3	63	0.13	61.02	0.13	1.26	0.58
t_1	21	0.55	4.21	0.31	0.23	5.46
t_1	42	0.26	492.85	0.01	0.98	0.33
t_1	63	1.42	OOT/M	OOT/M	OOT/M	OOT/M
t_2	21	0.50	8.22	1.50	0.57	2.63
t_2	42	1.18	OOT	OOT	OOT	OOT
t_3	21	0.39	20.03	0.96	0.80	9.67
t_3	42	0.96	OOT/M	OOT/M	OOT/M	OOT/M

Table 1: Incomplete features, expected link impact, and runtimes averaged over ten random seeds.

Implementation: Our implementation uses the CUDD package (Somenzi 1998) to manage ADDs and was implemented in C++.

Negative cycles in Floyd-Warshall are problematic because they cause the values across the matrix to slowly move toward $-\infty$, which results in a large number of leaf nodes in our ADDs. In our implementation we avoid this issue by incrementally checking for negative cycles during the execution of Floyd-Warshall and when a negative cycle is detected the values across the matrix for that model are immediately reduced to $-\infty$. This does not change the value of the final count because all models with negative cycles are inconsistent and once a negative cycle is introduced in Floyd-Warshall it is never removed by more information.

Experiments: We conduct three sets of experiments, all run on a 2 GHz Xeon processor. The first (with results presented in Table 1) introduces differing numbers of incomplete features so that we can measure the impact of incompleteness upon knowledge compilation (constructing the ADD ψ_∞), Floyd-Warshall, and model counting as plans increase in size and across the domains. The second, an ablation experiment, increases a single type of incomplete feature (precondition, add, or delete) with respect to a single plan to assess the total time to compute partial consistency. The third, in Table 2 provides runtime by domain.

In Table 1, we present the number of incomplete features ($|F|$) per instance (excluding the known preconditions and effects), the average expected number of incomplete features directly impacting a causal link ($E[F(l)]$), the knowledge compilation time to construct the ψ_∞ ADD ($T(\psi_\infty)$), the time to complete the symbolic Floyd-Warshall ($T(FW)$), and the time to complete model counting in the weak ($T(WC)$) and strong ($T(SC)$) cases; all time is in seconds. Instances surpassing a 15 minute limit are denoted "OOT", those surpassing a 2GB memory limit are denoted "OOM", and both on different random seeds are denoted "OOT/M". Table 2 lists results by domain for each domain in the 2011 IPC. Within each domain, we evaluate instances with 0, 9, 18, and 27 randomly inserted possible features F, and report the minimum, maximum, and average total runtime in seconds (from knowledge compilation, Floyd-Warshall, and both strong and weak consistency evaluation) across the plans for each instance in the domain and ten random seeds. The last column in Table 2 lists two numbers: the number of instance and random seed pairs under which the plan was evaluated within five minutes, and the total number of instances where an IPC planner generated a plan times the number of random seeds.

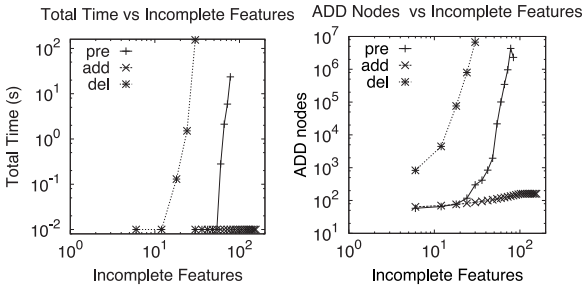

Figure 2: Time(s) and ψ_∞ ADD Nodes vs # of Features

Discussion: In the first experiment, we see that the expected number of incomplete features directly affecting a link $E[F(l)]$ is correlated with runtime across all phases of the algorithm. Knowledge compilation time $T(\psi_\infty)$ accounts for a majority of the runtime, followed by model counting for strong $T(SC)$ or weak $T(WC)$ partial consistency. By domain, the differences in runtime seem to be explained by the expected number of incomplete features per causal link, which means that our approach scales well in the size of the plans, but not as well in the density of the incompleteness. However, we note that an incomplete domain can consist of many incomplete features that do not affect the plan. There does not appear to be correlation between the domain and runtime, meaning that the degree of concurrency (which varied among the domains) may not play a large role (as supported by the runtimes for Floyd-Warshall $T(FW)$); however, it was possible to achieve lower runtimes for the higher expected number of incomplete features per link in Matchcellar because the plans were relatively smaller.

In our second experiment we independently increased the type of incomplete feature (preconditions, add, or delete effects) in the second Sokoban plan s_2; we saw similar results in other plans and domains. Figure 2 plots the total time for knowledge compilation, Floyd-Warshall, and strong partial

	$\|F\|$	Min (s)	Max (s)	Avg (s)	#Eval
Pegsol	0	0	0.03	0.01	200 / 200
	9	0	0.09	0.01	200 / 200
	18	0	0.87	0.04	200 / 200
	27	0.01	9.34	0.32	200 / 200
Matchcell	0	0	0.1	0.03	200 / 200
	9	0	3.74	0.18	200 / 200
	18	0	66.36	3.33	200 / 200
	27	0.01	299.96	22.61	200 / 200
Parking	0	0	0.54	0.15	200 / 200
	9	0.01	1.08	0.21	200 / 200
	18	0.01	2.14	0.36	200 / 200
	27	0.02	21.93	1.39	200 / 200
Sokoban	0	0.04	7	0.86	120 / 120
	9	0.05	16.46	1.09	120 / 120
	18	0.06	17.81	1.52	120 / 120
	27	0.07	110.24	4.65	120 / 120
Floortile	0	0.06	3.2	0.81	137 / 150
	9	0.07	247.06	9.55	137 / 150
	18	0.07	298.47	31.46	137 / 150
	27	0.06	293.19	46.15	137 / 150
Crewplan	0	0.04	6.76	1.34	200 / 200
	9	0.04	293.5	7.1	200 / 200
	18	0.04	295.18	19.79	200 / 200
	27	0.04	290.47	53.84	200 / 200
Storage	0	0.58	144.26	21.44	190 / 190
	9	0.62	287.82	29.95	190 / 190
	18	0.6	286.17	35.54	190 / 190
	27	0.62	269.03	34.83	190 / 190
TMS	0	2.14	214.3	67.55	77 / 190
	9	5.82	214.5	71.41	129 / 190
	18	3.65	212.92	63.07	85 / 190
	27	3.65	214.3	62.67	97 / 190
Elevators	0	18.11	291.39	114.71	106 / 200
	9	4.39	291.39	110.52	178 / 200
	18	4.39	291.39	123.74	184 / 200
	27	0	0	0	0 / 200
Trn&open	0	8.5	276.12	94.35	121 / 190
	9	8.55	276.12	87.93	94 / 190
	18	8.55	276.12	90.88	73 / 190
	27	8.55	243.89	85.35	73 / 190

Table 2: Results for all 2011 IPC plans: min, max, and avg. time in seconds across all plans in domain.

consistency model counting for each number of each feature; it also shows the number of ADD nodes in the representation of ψ_∞. We see that incomplete delete effects have the largest effect upon runtime, followed by incomplete preconditions. Additional add effects have little to no effect. Because runtime is dominated by knowledge compilation, we can explain the impact of each action feature in its effect upon ψ_∞. Possible delete effects and possible preconditions tend to increase the disjunction (c.f., the definitions of $nt(s_k, l)$ and $\psi(s_i)$, respectively) in ψ, while possible add effects do not (c.f., the definition of $\psi(l)$). As disjunction increases, the ADD representation of ψ can become large, which determines the cost of all of the ADD operations (which are polynomial in the ADD size).

In our third experiment (Table 2) we see that there are some significant differences between the domains. The domains with relatively low average evaluation time (and easily evaluable instances) include Pegsol, Matchcellar, Parking, Sokoban, Crewplan, and Storage. Those that are more challenging include Floortile, TMS, Elevators, and Turn and open. Interestingly, some of these domains are relatively insensitive to the number of incomplete features (over the range that we tested); for example, Elevators, TMS, and Turn and open all happen to be insensitive, but are from the relatively challenging group of domains. If we examine minimum, maximum, and average steps per plan in these domains (Figure 3), it appears that the difficulty stems from the plan size. However, from Table 1, we see that as incompleteness increases, the plan size has less of a relative impact on runtime.

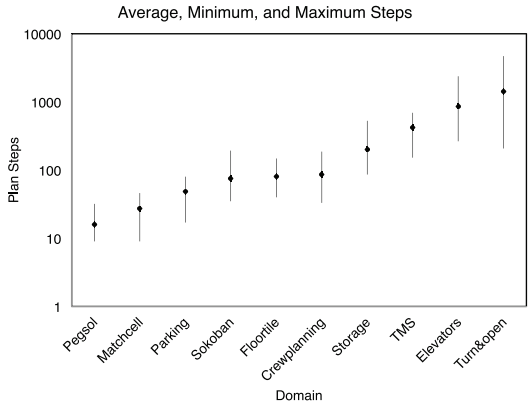

Figure 3: Avg., min, and max steps per plan per domain.

Related Work

Evaluating plans in incomplete domains was first considered by Garland and Lesh (2002). Weber and Bryce (2011) extend this work to synthesize plans that are robust to incompleteness, and show that counting diagnoses of failure to bias plan synthesis can outperform counting models, as studied by Nguyen, Kambhampati, and Do (2010). Weber and Bryce (2011) also showed that planning in incomplete domains can be formulated as conformant planning, in which partially ordered plan evaluation (as described herein) is known to be costly (Kushmerick, Hanks, and Weld 1994).

If we assume that the domain model is fully known, then the weak and strong partial consistency algorithms are solving a disjunctive temporal problem (Stergiou and Koubarakis 2000), and are similar to approaches to planning with disjunctive ordering constraints (Nguyen and Kambhampati 2001). The primary difference in our work is the disjunctive constraints are effected by incomplete knowledge of the domain rather than being unconditionally imposed as a means of threat resolution.

While we are not the first to study conditional temporal problems, we are (to our knowledge) the first to count solutions. Schwalb, Kask, and Dechter (1994) solve the corresponding satisfiability problem (i.e., finding a satisfying assignment to propositional variables, where an induced tem-

poral problem is consistent). Tsamardinos, Vidal, and Pollack (2003) study the weak, strong, and dynamic consistency of conditional temporal problems. Our work is a natural extension to these prior works and can be extended to consider probabilistic consistency through weighted model counting.

Sheini et al. (2005) solve very similar encodings of soft temporal constraints using Bender's decomposition to find models of Boolean constraints using a SAT solver and checking the implied temporal constraints using an STP/DTP solver. While the overall encoding resembles ours, the model counting nature of our work motivates a knowledge compilation approach rather than search.

Conclusion & Future Work

Model counting is a useful technique for evaluating the degree of weak or strong temporal consistency when a problem is not fully consistent. While we demonstrate our approach with PDDL 2.1 action models, our formulation of partial consistency can incorporate other models. Our approach combines knowledge compilation and symbolic constraint propagation to find consistent temporal problems. We then count the number of temporal problems that are consistent as model counting in ADDs. Our experiments show that our approach scales to evaluate large temporal plans and largely incomplete temporal planning domains.

As previously alluded, this work is a necessary step toward synthesizing plans in incomplete temporal planning domains. Practical planning algorithms will evaluate partial plans inside of a search algorithm and it is likely that approximate plan evaluation, as well as search heuristics, will be required to scale-up. We intend to address these issues in future work by extending approaches taken in classical (non-temporal) planning (Weber and Bryce 2011).

We also plan to extend this work to handle unknown action durations, which can be formulated as an extension to temporal controllability. Our setting is most similar to probabilistic STPs (Tsamardinos, Pollack, and Ramakrishnan 2003) that maximize the probability that uncontrollable time points can be legally executed, except that we do not assume knowledge about probability distributions. We foresee that uncertain duration actions will require us to extend the CTN model so that the propositional models imply problems similar to probabilistic STPs instead of just STPs.

Acknowledgements: This material is based upon work supported by DARPA under Contract No. HR0011-07-C-0060 and Utah State University.

References

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*, 188–191.

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. McGraw-Hill.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Fox, M., and Long, D. 2003. PDDL2.1: An extension of PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Garland, A., and Lesh, N. 2002. Plan evaluation with incomplete action descriptions. In *Proceedings of Eighteenth National Conference on Artificial Intelligence*, 461–467.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2009. Model counting. In *Handbook of Satisfiability*. 633–654.

Kushmerick, N.; Hanks, S.; and Weld, D. S. 1994. An algorithm for probabilistic least-commitment planning. In *Proceedings of the 12th National Conference on Artificial Intelligence*, 1073–1078.

Nguyen, X., and Kambhampati, S. 2001. Reviving partial order planning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 459–466.

Nguyen, T. A.; Kambhampati, S.; and Do, M. B. 2010. Assessing and generating robust plans with partial domain models. In *ICAPS-10 Workshop on Planning and Scheduling Under Uncertainty*.

Schwalb, E.; Kask, K.; and Dechter, R. 1994. Temporal reasoning with constraints on fluents and events. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1067–1072.

Sheini, H. M.; Peintner, B.; Sakallah, K. A.; and Pollack, M. E. 2005. On solving soft temporal constraints using SAT techniques. In *Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming*, 607–621.

Simpson, R. M.; Kitchen, D. E.; and McCluskey, T. L. 2007. Planning domain definition using GIPO. *The Knowledge Engineering Review* 22(02):117–134.

Somenzi, F. 1998. *CUDD: CU Decision Diagram Package Release 2.3.0*. University of Colorado at Boulder.

Stergiou, K., and Koubarakis, M. 2000. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence* 120(1):81 – 117.

Tsamardinos, I.; Pollack, M. E.; and Ramakrishnan, S. 2003. Assessing the probability of legal execution of plans with temporal uncertainty. In the ICAPS-03 Workshop on Planning and Scheduling under Uncertainty.

Tsamardinos, I.; Vidal, T.; and Pollack, M. E. 2003. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints* 8(4):365–388.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE 2.0: An integrated tool for designing planning domains. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, 336–343.

Weber, C., and Bryce, D. 2011. Planning and acting in incomplete domains. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling*, 274–281.

Wu, K.; Yang, Q.; and Jiang, Y. 2007. ARMS: An automatic knowledge engineering tool for learning action models for AI planning. *Knowledge Engineering Review* 22(2):135–152.