Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling

Chance-Constrained Consistency for Probabilistic Temporal Plan Networks

Pedro H.R.Q.A. Santana and **Brian C. Williams**
Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, MERS
32 Vassar St., Room 32-224, Cambridge, MA 02139, {psantana,williams}@mit.edu

Abstract

Unmanned deep-sea and planetary vehicles operate in highly uncertain environments. Autonomous agents often are not adopted in these domains due to the risk of mission failure, and loss of vehicles. Prior work on contingent plan execution addresses this issue by placing bounds on uncertain variables and by providing consistency guarantees for a 'worst-case' analysis, which tends to be too conservative for real-world applications. In this work, we unify features from trajectory optimization through risk-sensitive execution methods and high-level, contingent plan execution in order to extend existing guarantees of consistency for conditional plans to a chance-constrained setting. The result is a set of efficient algorithms for computing plan execution policies with explicit bounds on the risk of failure. To accomplish this, we introduce Probabilistic Temporal Plan Network (pTPN), which improve previous formulations, by incorporating probabilistic uncertainty and chance-constraints into the plan representation. We then introduce a novel method to the chance-constrained strong consistency problem, by leveraging a conflict-directed approach that searches for an execution policy that maximizes reward while meeting the risk constraint. Experimental results indicate that our approach for computing strongly consistent policies has an average scalability gain of about one order of magnitude, when compared to current methods based on chronological search.

1 Introduction

Real-world environments are inherently uncertain, causing agents to inevitably experience some level of risk of failure when trying to achieve their goals. Failure is not restricted to the operation of autonomous systems in hazardous environments, but an integral part of life. For example, an unexpected flat tire might incur a lot of extra delay in a usually quick commute to work. Instead of neglecting the existence of risk or overlooking the fact that unexpected things might have important impacts on a mission, it becomes key for autonomous systems trusted with critical missions in real-world situations to have a keen sensitivity to risk and to be able to incorporate uncertainty into their decision-making.

This work addresses the problem of extracting execution policies with risk guarantees from contingent plans with uncertainty. It drew great inspiration from interactions with groups in the Woods Hole Oceanographic Institution (WHOI) and NASA's Jet Propulsion Laboratory (JPL), who share the responsibility of planning and operating multimillion dollar missions in extreme environments (deep sea and planetary exploration) with very stringent safety requirements. The current practice for ensuring safety in these missions requires groups of engineers to reason over a very large number of potential decisions and execution scenarios that might unfold during execution, which is a challenging, time-consuming, and error prone process.

There has been significant effort in the scientific community to help answer the question of whether a feasible execution policy exists in the presence of uncontrollable contingencies in the plan. Among the most important contributions, (Tsamardinos, Vidal, and Pollack 2003; Effinger et al. 2009; Venable et al. 2010; Hunsberger, Posenato, and Combi 2012) extend the notions of strong, dynamic, and weak controllability (also referred as consistency) originally used in the context of temporal uncertainty (Vidal and Ghallab 1996; Vidal 1999; Morris, Muscettola, and Vidal 2001) to uncontrollable contingencies. They also present tests that guarantee the existence of feasible solutions under different levels of information about the uncertainty in the plan. These consistency tests, however, all have the caveat of representing uncertainty as set-bounded quantities, i.e., as intervals of values with no associated probability distribution. In other to guarantee feasibility in all possible scenarios, consistency-checking algorithms based on set-bounded uncertainty end up performing a worst-case analysis. When considering situations where uncertainty causes small plan deviations around otherwise "nominal" values, these set-bounded consistency criteria work well and output robust, albeit conservative, execution policies. Nevertheless, they have difficulties handling problem instances where uncertainty can potentially lead the system to very hard or even irrecoverable scenarios, often returning that no robust execution policy exists. This is most certainly undesirable, since reasonable amounts of risk can usually be tolerated for the sake of not having the autonomous agent sit idly due to its absolute "fear" of the worst.

Given a description of a contingent plan, this work improves on the previously cited literature on conditional plan execution by extending the notions of weak and strong plan consistency to a risk-bounded setting and providing effi-

cient algorithms for determining (or refuting) them. These risk bounds are also known as chance-constraints (Birge and Louveaux 1997). Weak and strong consistency are useful concepts when planning missions for agents whose embedded hardware has very limited computation and telecommunication power, making it hard for them to come up with solutions 'on the fly' or for remote operators to intervene in a timely fashion. Chance-constrained weak consistency (CCWC) is a useful concept for missions where agents operate in static or slow changing environments after an initial scouting mission aimed at reducing plan uncertainty. Chance-constrained strong consistency (CCSC), on the other hand, removes the need for a scouting mission and tries to determine the existence of a solution that, with probability greater than some threshold, will succeed irrespective of the outcomes of uncertainty in the plan. Strong consistency is clearly more conservative, but it is appealing to mission managers because strongly consistent policies require little to no onboard sensing and decision making, greatly reducing the agents' complexity and costs. They also reduce or completely eliminate the need to coordinate between multiple agents. Finally, the robustness of a strongly consistent policy makes it easier to check by human operators before it is approved for upload to the remote agent.

We introduce Probabilistic Temporal Plan Networks (pTPNs) as our representation of contingent plans. Our pTPN representation holds a lot of similarities with Temporal Plan Networks with Uncertainty (TPNU) (Effinger et al. 2009; Effinger 2012), Conditional Temporal Plans (CTPs)(Tsamardinos, Vidal, and Pollack 2003), Disjunctive Temporal Problems with Uncertainty (DTPU)(Venable et al. 2010), and the Conditional Simple Temporal Network with Uncertainty (CSTNU) (Hunsberger, Posenato, and Combi 2012), but extends them in two important ways. First, pTPNs allow uncontrollable choices (discrete, finite domain random variables) to have their joint distributions described by a probability model, as opposed to a purely set-bounded uncertainty representation in DTPUs and CTPs. Second, pTPNs allow the user to specify admissible risk thresholds that upper bound the probability of violating sets of constraints in the plan, a missing feature in CTPs, DTPUs, and TPNUs. The latter extension in an important improvement of pTPNs over previous representations when modeling many real-world problems, where risk-free plans that are robust to all possible uncontrollable scenarios are often unachievable. Our pTPNs are compiled from contingent plans descriptions in RMPL (Williams et al. 2003; Effinger et al. 2009), but other forms of generating contingent plans are equally amenable to our methods (Drummond, Bresina, and Swanson 1994; Dearden et al. 2003).

Our algorithms reason quantitatively about the probability of different random scenarios and explore the space of feasible solutions efficiently while bounding the risk of failure of an execution policy below a user-given admissible threshold. While state-of-the-art methods in the conditional and stochastic CSP literature rely on a combination of chronological (depth-first) search and inference in the space of contingencies in order to quickly find satisficing solutions (Fargier et al. 1995; Fargier, Lang, and Schiex 1996; Stergiou and Koubarakis 2000; Gelle and Sabin 2006; Tarim, Manandhar, and Walsh 2006), in this work we introduce a "diagnostic" approach based on Conflict-Directed A^* (CDA*)(Williams and Ragno 2007). By continuously learning subsets of conflicting constraints and generalizing them to a potentially much larger set of pathological scenarios, our algorithms can effectively explore the space of robust policies in best-first order of risk while ensuring that it is within the user-specified bound. For the problem of extracting a strongly consistent policy from a contingent plan description, our numerical results showed significant gains in scalability for our approach.

This work is organized as follows. Section 2 presents a simple example motivating our methods, followed by formal definitions of pTPNs and our notions of chance-constrained consistency in Section 3. Next, Section 4 presents algorithms for determining chance-constrained weak and strong consistency of pTPNs. The numerical results in Section 5 indicate that our framing of the problem of determining CCSC outperforms the current practice of using chronological search, followed by our final conclusions in Section 6.

2 Approach in a nutshell

Here we motivate the usefulness of chance-constrained consistency on a very simple commute problem, whose pTPN representation is given in Figure 1. We start at home and our goal is to be at work for a meeting in at most 30 minutes. Circles represent the start and end events of temporally-constrained activities called episodes. Simple temporal constraints (Dechter, Meiri, and Pearl 1991) are represented by arcs connecting temporal events and represent linear bounds on their temporal distance. For simplicity, we assume that we are given only three possible choices in this pTPN: we can either ride a bike to work, drive our car, or stay home and call our employer saying that we will not be able to make it to work today. The rewards (R values) associated with each one of these choices in Figure 1 correspond to how much money we would make that day minus the cost of transportation. Uncontrollable choices are depicted in Figure 1 by double circles with dashed lines. These are random, discrete events that affect our plan and whose probability model is also given in Figure 1.

In this example, the uncontrollable choices model what might "go wrong" during plan execution and the impact of these unexpected events on the overall duration of the plan. For example, if we decide to ride a bike to work (the option with the highest reward), there is the possibility that we might slip and fall. This event has a minor effect on the duration of the ride, but would force us to change clothes at our workplace because we cannot spend the day in a dirty suit. Since we only have 30 minutes before the meeting starts, the uncontrollable event of slipping would cause the overall plan to be infeasible. A similar situation happens if we choose to drive our car and happen to be involved in an accident.

By ignoring probabilities and using a consistency checker for the pTPN in Figure 1 based on a set-bounded representation of uncertainty, we would realize that the pTPN is guaranteed to be consistent. Unfortunately, unless we had a way of telling ahead of time whether we would slip from the bike

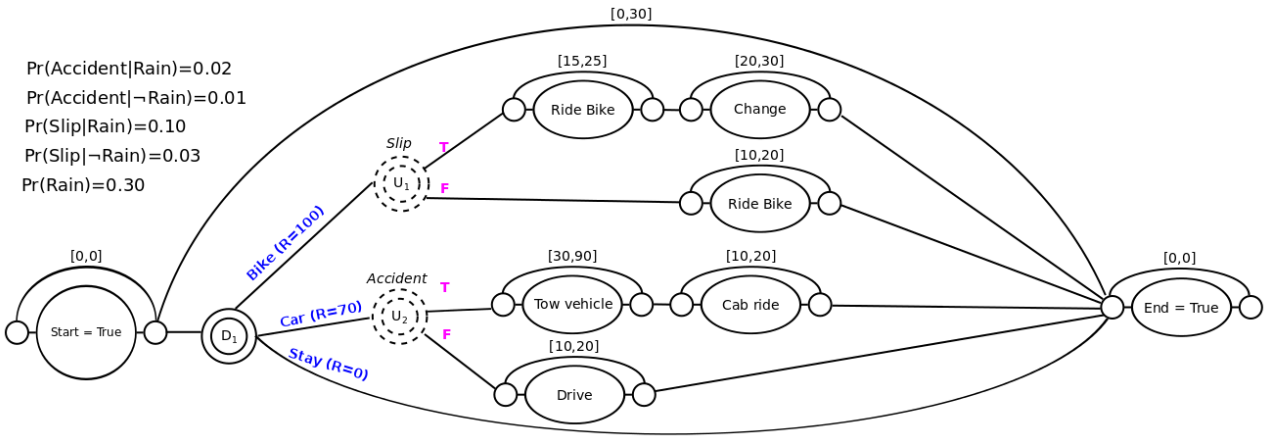

Figure 1: A pTPN for a simple plan to get to work

or be in a car accident, the suggested policy would be to always stay home! This is because, for the choice of riding a bike or driving our car to work, there are uncontrollable scenarios that cause the plan to fail, causing the set-bounded consistency checker to fall back to the safe, albeit undesirable, choice of staying at home. This clearly disagrees with our common sense, since people try to achieve their goals while acknowledging that uncontrollable events might cause them to fail. Next, we show how our chance-constrained approach would produce execution policies that agree with what we would expect a "reasonable" agent to do.

Let's consider the case where we accept that our plan might fail, as long as the risk Δ is no more than 2%. Given that riding a bike is the option with the highest reward, our algorithm would deem bike ridding the most promising and would start by checking if choosing to ride a bike meets the chance-constraint $\Delta \leq 2\%$. If there existed a feasible activity schedule satisfying the temporal constraints for both values of $Slip$, we could pick this schedule and our risk of failure would be zero, which is clearly less than our risk bound. However, our algorithm concludes that the scenario $Slip = True$ is inconsistent with the overall temporal constraint of arriving at the meeting in less than 30 minutes, so there must exist a nonzero risk of failure in this case. According to the model in Figure 1, the probability of having slip is $\Pr(Slip) = 5.1\%$, so riding a bike does not meet the chance-constraint $\Delta \leq 2\%$. The next best option is riding a car, where now we are subject to the uncontrollable event of being in a car accident. Following a similar analysis, we conclude that the risk of our plan being infeasible in this case is $\Pr(Accident) = 1.3\%$, which meets the chance-constraint. Therefore, our algorithm would advise us to drive to work within the temporal bounds shown in Figure 1 for the case where no accident happens. We claim that this chance-constrained type of reasoning approximates the decision making process of mission managers much better than its set-bounded alternative, since operators need to commit to plans with acceptable levels of risk in order to extract some useful output from the remote explorer.

It is worthwhile to notice that choosing $\Delta < 1.3$ would have made staying at home the only feasible alternative. Hence, as in the set bounded approach, a chance constraint may still be too conservative to allow for a feasible solution. Moreover, if the overall temporal constraint of 30 minutes in Figure 1 were relaxed to 35 minutes, our algorithm would have been capable of finding a risk free scheduling policy for its first choice of riding a bike. This is because, in this case, there would exist a feasible schedule satisfying all temporal constraints on the upper side of the pTPN, i.e., temporal constraints activated by both $Slip = True$ and $Slip = False$.

This example highlights a few key elements of our approach. First, we divide the problem into generating a candidate "plan" as an assignment to the controllable choices and testing the plan against the chance constraints. Second, candidates are enumerated in best-first order based on reward. Third, testing feasibility of a chance-constraint is a fundamental task, in which estimating risk is costly. We frame risk estimation as a process of enumerating the most likely sets of scenarios that incur and do not incur risk (called conflicts and kernels, respectively). We observe that this can be formulated as a symptom-directed, "diagnostic" process, allowing us to leverage an efficient, conflict-directed best-first enumeration algorithm, Conflict-Directed A^* (CDA*)(Williams and Ragno 2007), to generate kernels and conflicts, and hence feasible and infeasible scenarios.

3 Problem statement

The previous section motivated the need to explicitly reason about risk and uncertainty when trusting autonomous systems with critical missions in hazardous environments. The outcome of this reasoning is an execution policy that not only performs well in terms of reward, but also keeps the risk of failure bounded below some acceptable threshold chosen by the mission manager. In this section, we make these notions precise.

3.1 Probabilistic Temporal Plan Networks

Recall that our objective is to robustly execute time critical missions, while being responsive to the environment

and providing a bound on the risk of failure. A TPN (Kim, Williams, and Abramson 2001; Walcott 2004; Effinger et al. 2009) as we define it here is a temporal plan that includes contingencies and sensing actions. Temporal plans encode mission activities and mission critical timing constraints between these activities. Conditions on sensing actions improve robustness, by enabling the plan to tailor its course of action based on environment conditions. Contingencies improve robustness by offering alternative courses of action with varying requirements, such as constraints on duration, utility and resources employed. Sensing actions and contingencies are represented, respectively, as uncontrollable and controllable choices.

A probabilistic TPN models uncertainty by representing exogenous inputs as random variables, and specifies requirements on risk of failure in terms of chance constraints over subsets of the TPN. This is in contrast to a TPNU (Effinger et al. 2009), which specifies interval bounds over possible values of exogenous variables, and requires that a consistent execution of the TPNU is feasible for all possible assignments to exogenous variables. An example graphical representation of a pTPN was previously given in Figure 1. Formally, a pTPN is defined as follows:

Definition 1 (pTPN)**.** *A Probabilistic Temporal Plan Network (pTPN) is a tuple $P = \langle \mathbb{E}, \mathbb{T}, \mathbb{D}, \mathbb{U}, R, P, \mathbb{CH} \rangle$, where $\mathbb{E}$ is the set of episodes; $\mathbb{T}$ is the set of temporal constraints between episodes in the plan; $\mathbb{D} = \{D_1, \ldots, D_N\}$ is the set of N controllable choices (also called decisions), each D_i being a discrete, finite domain variable with domain $Dom(D_i)$; $\mathbb{U} = \{U_1, \ldots, U_M\}$ is the set of M uncontrollable choices, each U_i being a discrete, finite domain random variable with domain $Dom(U_i)$; $R : \mathbb{D} \rightarrow \mathbb{R}$ is a reward function that maps from decisions to values of reward; $P : U \rightarrow [0, 1]$ is a probability model that maps from joint realizations of uncontrollable choices to probability values. $\mathbb{CH}$ is the set of chance-constraints defined over sets of constraints in the plan.*

Episodes in a pTPN model activities that have to be performed, such as "Drive" in Figure 1. They are defined as follows:

Definition 2 (Episode)**.** *An episode is a tuple $E = \langle e_s, e_e, A, SC, G \rangle$, where $e_s, e_e \in \mathbb{R}$ are the episode's start and end events, respectively*[1]*; A is an activity to be performed during the period of time $v_e - v_s$; SC is a set of state constraints that should hold during $e_e - e_s$, where "state" refers to the internal state of the system that is being controlled during plan execution; and G is a guard condition that activates the constraints in the episode.*

As previously mentioned, one important feature of pTPNs is the possibility of specifying admissible risk thresholds for the violation of plan constraints. These risk thresholds are called chance-constraints (Birge and Louveaux 1997).

The combination of assignments to controllable and uncontrollable choices defines which constraints in the pTPN should be satisfied by our scheduling policy. This motivates our definition of controllable and uncontrollable scenarios.

[1]Time values with respect to a temporal reference $e_{ref} = 0$.

Definition 3 (Controllable (Uncontrollable) scenario)**.** *A controllable (uncontrollable) scenario $CS \in \mathbb{CS}$ ($US \in \mathbb{US}$) corresponds to a full assignment $D{=}d$ ($U{=}u$) to the controllable (uncontrollable) choices in the pTPN.*

As seen in Figure 1, the choices in a pTPN define a "path" of episodes and constraints that should be taken into account in the plan. This path is encoded in the pTPN by the guard conditions for each episode. Similar to (Effinger 2006), we frame the problem of finding a feasible plan in a pTPN as solving a Constraint Optimization Problem (COP):

Definition 4 (pTPN as COP)**.** *Let a pTPN be defined as before. Its equivalent COP is given by:*

- *Variables $\mathbb{V} = \mathbb{V}_{co} \cup \mathbb{V}_{ch} \cup V_I$, $\mathbb{V}_{co} \cap \mathbb{V}_{ch} = \emptyset$, where $\mathbb{V}_{co}$ is the set of constraint variables and $\mathbb{V}_{ch} = \mathbb{D} \cup \mathbb{U}$ is the set of choice variables (controllable and uncontrollable). Every $V \in \mathbb{V}$ has domain $Dom(V)$;*
- *Initial variable V_I, $Dom(V_I) = True$;*
- *Activity constraints $\mathbb{C}_A$, where $C_A \in \mathbb{C}_A$ is of the form $C_A : \{V_{ch} = v\} \rightarrow Active(\mathbb{V}')$, with $V_{ch} \in \mathbb{V}_{ch}$, $\mathbb{V}' \subseteq \mathbb{V}$;*
- *Compatibility constraints $\mathbb{C}_c = \bigcup_i SC_i$, where SC_i are the constraints for the $i-th$ episode. A compatibility constraint only needs to be satisfied if all of its variables are active;*
- *Chance constraints $\mathbb{CH}$ defined over sets of constraints in the plan;*
- *Reward function $R : \mathbb{D} \rightarrow \mathbb{R}$ mapping from decisions to values of reward;*
- *Probability model $P : U \rightarrow [0, 1]$ mapping from joint realizations of uncontrollable choices to probability values.*

In this work, we will focus on the case where there is a single overall chance-constraint with admissible risk Δ over the complete set of temporal constraints in the plan. In other words, this chance-constraint upper bounds the probability of violating any of the temporal constraints in our plan.

3.2 Chance-constrained consistency

This section extends the "risk-averse" notions of weak and strong consistency from (Vidal and Ghallab 1996; Tsamardinos, Vidal, and Pollack 2003; Effinger et al. 2009) to a setting where solutions involving some level of risk are accepted. As previously mentioned, allowing plans to contain reasonable levels of risk is usually required for systems operating under uncertainty. If no risk is allowed, a robust execution strategy will hardly ever exist.

Intuitively, a chance-constrained weakly consistent pTPN is one that, for every possible uncontrollable scenario within a set $\mathbb{US}'$, there exists at least one controllable scenario CS so as to yield a feasible CSP. Therefore, given knowledge about which uncontrollable scenario is true, we can always choose CS such that a solution to the CSP exists. In our chance-constrained formulation, we guarantee that the risk of the true uncontrollable scenario being outside of the feasible set $\mathbb{US}'$ is less or equal to Δ. Strong consistency, on the other hand, requires that at least one controllable scenario

CS exists such that there exists at least one common solution to all CSP's generated by all uncontrollable scenarios $US \in \mathbb{US}'$. Once again, our chance-constrained formulation guarantees that our strongly consistent policy will be feasible with probability at least $1-\Delta$. While weak consistency requires complete prior knowledge of the true uncontrollable scenario prior to choosing CS, strong consistency requires none: we could just "close our eyes" and pick a common solution to all CSP's while completely disregarding the uncontrollable scenario being unfolded.

Let $CSP \leftarrow Ac(pTPN, CS, US)$ be a function that returns a CSP formed by all active constraints from a pTPN given a controllable scenario CS and an uncontrollable one US. Also, let $sols \leftarrow Sol(CSP)$ by an algorithm that is able to return a set of possible solutions (the absence of solutions is denoted by $sols=\emptyset$) to a CSP[2]. We are now ready to define the notions of chance-constrained weak and strong consistency.

Definition 5 (Chance-constrained weak consistency (CCWC))**.** *A pTPN is said to be CCWC with risk Δ iff there exists a set of uncontrollable scenarios $\mathbb{US}' \subseteq \mathbb{US}$, where $\Pr(\mathbb{US}') > 1-\Delta$, such that $\forall US \in \mathbb{US}'$, there exists a controllable scenario CS such that $Sol(Ac(pTPN, CS, US)) \neq \emptyset$.*

Definition 6 (Chance-constrained strong consistency (CCSC))**.** *A pTPN is said to be CCSC with risk Δ iff there exists a controllable scenario CS and a set of uncontrollable scenarios $\mathbb{US}' \subseteq \mathbb{US}$, where $\Pr(\mathbb{US}') > 1-\Delta$, such that*

$$\bigcap_{US_i \in \mathbb{US}'} Sol(Ac(pTPN, CS, US_i)) \neq \emptyset. \quad (1)$$

Definitions 5 and 6 become equivalent to the standard definitions of weak and strong consistency if we choose $\Delta = 0$, i.e., no risk. Moreover, they formally define the concepts of weak and strong chance-constrained consistency in terms of a subset of uncontrollable scenarios $\mathbb{US}' \subseteq \mathbb{US}$ with appropriate probability and feasible assignments to the controllable choices in the plan. Determining those assignments is the topic of the subsequent sections.

4 Chance-constrained consistency of pTPNs

Our goal in determining chance-constrained weak and strong consistency as defined in Section 3.2 was to provide the user with an execution policy that maximizes plan reward while keeping the risk of failure bounded by Δ. The execution policy consists of (I) an assignment to the controllable choices in the plan, plus (II) a set of consistent schedules, represented by a Simple Temporal Network[3] (STN) (Dechter, Meiri, and Pearl 1991) with a risk of failure upper bounded by $\Delta \in [0, 1]$.

The conditional nature of temporal constraints on a pTPN, which might be activated by different combinations of assignments to controllable and uncontrollable choices, makes the evaluation of chance-constraints challenging when constructing a solution to (I) and (II). Hence, as per our example in Section 2, here we adopt a "generate and test" approach to solving (I) and (II), where (I) corresponds to the "generate" step and (II) to "test". More precisely, the first step in our algorithms enumerates assignments to controllable choices in best-first order so that the most promising solutions in terms of reward are checked for feasibility first. Once (I) is fixed, step (II) then answers the question of whether it is possible to find a feasible STN with probability at least $1 - \Delta$.

The algorithms in Sections 4.2 and 4.3 frame (I) as a combinatorial optimization problem and solve it using OpSAT (Williams and Ragno 2007):

1. **Decision variables**: Controllable choices $\mathbb{D}$;
2. **Objective function**: $\max R$, the reward function defined over $\mathbb{D}$ described in Definition 1;
3. **Constraints**: Step (II) returns a feasible solution.

Throughout the algorithms in this section, the function $NextBest(\cdot)$ is implemented using CDA* in order to enumerate candidate solutions in best-first order while taking previously discovered conflicts into account. A very detailed discussion of how to perform this enumeration can be found in (Williams and Ragno 2007).

Evaluating (II) is a key step in our algorithms, specially for chance-constrained strong consistency. In fact, one important contribution of this paper is the framing of chance-constraint evaluation for strongly consistent plans as a discrete "diagnostic" process, where "symptoms" correspond to inconsistencies in temporal constraints and "diagnoses" are pathological uncontrollable scenarios. Recalling our example in Section 2, the pathological scenarios for an overall temporal constraint of 30 minutes were $Slip = True$ if we chose to ride a bike to work, or $Accident = True$ if we chose to drive a car. Step (II) is handled differently in weak and strong consistency, so its explanation is deferred to later sections.

Section 4.1 introduces a labeled constraint representation of Definition 4, an encoding that makes explicit the connection between each temporal constraint and the choices that activate it, while eliminating intermediate activation constraints. Next, Section 4.2 presents an algorithm for checking chance-constrained weak consistency of a pTPN and its corresponding mapping from uncontrollable scenarios to corresponding feasible assignments to the controllable choices. We then proceed to the formulation of chance-constrained strong consistency as a "diagnosis" problem solved by CDA* in Section 4.3.

4.1 Labeled constraint representation of a pTPN

Similar to (Tsamardinos, Vidal, and Pollack 2003), our algorithms use an internal representation of pTPNs as labeled

[2]There are various methods available in the literature for checking consistency of simple temporal networks, whether they are fully controllable STNs (Dechter, Meiri, and Pearl 1991) or STNs with Uncertainty (STNUs) (Vidal 1999), the latter admitting the existence of uncontrollable activity durations. Checking temporal consistency of STNs and STNUs is not the focus of this work, so we treat $Sol(\cdot)$ as a black box.

[3]Simply stated, a consistent STN consists of a conjunction of simple temporal constraints that can be jointly satisfied.

constraints for efficient determination of the active plan constraints under different controllable and uncontrollable scenarios. Intuitively, a constraint label is a complete description of the set of choices leading to the activation of a given constraint in the plan, as seen in Figure 1. We define label and labeled constraint as follows:

Definition 7 (Label)**.** *A label L denotes a logical expression written in Disjunctive Normal Form (DNF), i.e., $L{=}a_1 \vee \ldots \vee a_n$. The term a_i represents the i-th set of implicants of the constraint (De Kleer 1986) in the form of assignments to choice variables.*

Definition 8 (Labeled constraint)**.** *A labeled constraint $LC{=}\langle L, C\rangle$ consists of a label L and a constraint C such that*

$$L \Rightarrow C. \tag{2}$$

We say that LC is active iff L is true.

Definition 9 (Constraint)**.** *A constraint is a pair $C{=}\langle Sc, Re\rangle$, where Sc represents the constraint's scope (the set of variables involved in the constraint) and Re is a relation between the variables in Sc.*

For example, a simple temporal constraint between events v_i and v_j can be written as $Sc{=}\{v_i, v_j\}$ and Re: $lb{\leq}v_i{-}v_j{\leq}ub$, where ub and lb correspond, respectively, to the upper and lower bounds on the temporal distance between the events. As a grounded example, we could extract the following labeled constraints from the pTPN in Figure 1:

$$\begin{aligned}&(D_1{=}Car) \wedge (U_2{=}F) \Rightarrow \textit{Drive} \in [10, 20],\\ &\text{True} \Rightarrow \textit{Complete} \in [0, 30].\end{aligned} \tag{3}$$

The bottom labeled constraint in (3) has a "True" label because the plan has to be completed within 30 minutes, no matter the outcomes of choices in the plan. For the purposes of the forthcoming sections, we assume the availability of a simple function $LC \leftarrow toLC(pTPN)$ capable of converting a pTPN into a list of labeled constraints.

4.2 Chance-constrained weak consistency

Determining weak consistency is special in the sense that we are allowed to assume that the true uncontrollable scenario will be revealed to the agent before it has to make any decisions. This is what happens, for example, when we use a smart phone to determine the state of traffic before committing to a specific route to some destination. By doing so, we implicitly assume that the state of the roads will not change during the trip. Since the agent knows which uncontrollable scenario is the true one, the problem is no longer stochastic and the agent can pick the best possible assignment to the controllable choices in terms of reward for that specific uncontrollable scenario. This is exactly the process that Algorithm 1 reproduces.

Algorithm 1 starts by creating an OpSAT instance for reward maximization (Line 2). For each possible uncontrollable scenario (US), Algorithm 1 searches for assignments to the controllable choices ($candCS$) that maximize reward while yielding a feasible set of constraints. If it is able to find it, it adds US to the set of consistent scenarios (Line 11) and marks CS as the optimal assignment to controllable choices given US (Line 12). However, if the search algorithm runs out of candidate assignments to controllable choices, it marks US as being inconsistent (Line 7). If the probability of inconsistent scenarios ever violates the overall chance-constraint Δ, we are guaranteed that the plan is not weakly consistent within the given risk bound. Similarly, if the probability of consistent scenarios surpasses $1{-}\Delta$, we can return the optimal mapping from uncontrollable to controllable scenarios. It is worthwhile to notice that uncontrollable scenarios are always disjoint, so computing probabilities in Lines 8 and 13 consists of just adding together the probability for each scenario.

Algorithm 1 Chance-constrained weak consistency.

Input: pTPN, risk threshold Δ.
Output: Mapping $UtoC$ from uncontrol. to control. scenarios.

```
1: UtoC←∅, conUS←∅, incUS ← ∅
2: OpS←RewardMax(pTPN)
3: for US ∈ 𝕌𝕊 do
4:     cConf ← ∅, nextUS←False
5:     while not nextUS do
6:         if (candCS ← NextBest(cConf, OpS))==∅ then
7:             incUS ← incUS ∪ US, nextUS←True
8:             if Pr(incUS)>Δ then return FAIL
9:         else
10:            if Ac(pTPN, candCS, US) is consistent then
11:                conUS←conUS∪US, nextUS←True
12:                UtoC[US] ← candCS
13:                if Pr(conUS)≥1−Δ then return UtoC
14:            else
15:                cConf←LearnConflict(cConf, candCS)
16: return UtoC
```

Revealing all uncertainty associated with a plan might be as hard and costly as the plan itself, so determining CCWC might be of limited use in some cases. In situations where the cost of sensing and coordination between agents is considerable, it might be better to extract a policy from the plan description that is robust to a large enough fraction of the uncertainty in the plan. This is the topic of the next section.

4.3 Chance-constrained strong consistency

Differently from weak consistency, which considers the feasibility of each uncontrollable scenario separately, Definition 6 for strong consistency connects all uncontrollable scenarios $US \subseteq \mathbb{US}'$ together by requiring that they share a common solution to the CSPs defined by their sets of active constraints. Thus, evaluating the chance-constraint for a strongly consistent policy becomes a challenging and key subproblem, since we now have to consider how different subsets of constraints become active and interact across many possible different realizations of the uncontrollable choices. In weak consistency, the given uncontrollable scenario clearly revealed which constraints dependent on uncontrollable choices could be active in the plan. In the case of strong consistency, the subset of temporal constraints that must be satisfied to meet the chance-constraint is not known a priori, and finding it is computationally difficult.

Recall from the introduction of Section 4 that we pursue a "generate and test" approach in order to compute chance-constrained strongly consistent policies that maximize reward. As previously mentioned, step (I) in this approach is to choose a promising assignment CS to the controllable choices in terms of reward. Once the assignment is made, it separates the list of labeled constraints from the pTPN into three mutually exclusive groups defined in terms of the probability of activation of their labels:

- **Necessarily Active (NA)**: composed of all labeled constraints such that $\Pr(L_i|CS)=1$, which includes True labels (a True label will always be active). This is the set of all labels L_i that CS made True;
- **Necessarily Inactive (NI)**: composed of all labeled constraints such that $\Pr(L_i|CS)=0$. This is the set of all labels L_i that CS made False. According to Definition 8, these constraints are all inactive and do not influence plan feasibility.
- **Potentially Active (PA)**: the set of all remaining labeled constraints, for which $0 < \Pr(L_i|CS) < 1$. This is the set of all labels L_i containing assignments to uncontrollable choices that have not been made False by CS.

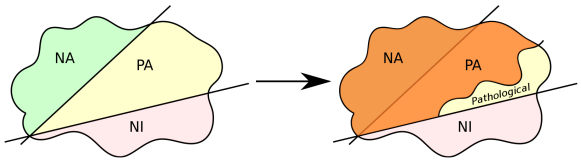

Figure 2: Partition of the constraints induced by an assignment to the controllable choices.

This split of the labeled constraints is schematically depicted on the left side of Figure 2. Given this partition, the subsequent evaluation of the chance-constraint can be graphically represented by the right side of Figure 2. In this figure, our algorithm is trying to find a subset of the temporal constraints in $NA \cup PA$ that is consistent and has probability greater than $1 - \Delta$ of containing the true set of active constraints for any possible uncontrollable scenario. If it succeeds, the set of schedules corresponding to step (II) of our "generate and test" approach can be computed from the STN composed of the temporal constraints contained in this subset. However, this is not an easy task to accomplish, since we are required that this subset of constraints is always consistent in order for a schedule to exist. Our chance-constraint then becomes important, since it allows temporal infeasibility to be resolved by dropping constraints activated by uncontrollable choices. This is done at the expense of increasing the risk of the resulting schedule being inconsistent in a number of different uncontrollable scenarios. The larger the admissible risk bound Δ, the more constraints can be dropped. It is easy to see that this subset of constraints must always cover all of *NA* in order for a strongly consistent schedule to exist. Also, no effort should be spent trying to cover any portion of *NI*.

It is usually the case that infeasibility with respect to the constraints in $NA \cup PA$ only manifests itself in a handful of scenarios that activate constraints that are hard to satisfy. Hence, one key contribution from this work is to frame the evaluation of the chance-constraint as a diagnostic process. The key insight behind the use of a conflict-directed method is to be able to quickly isolate constraints causing infeasibility of the STN and evaluate whether they incur a significant amount of risk. The numerical results in Section 5 show that our conflict-directed approach is able to detect violation of the chance-constraint Δ more efficiently than prior art based on chronological search methods.

Similar to step (I), we frame the problem of finding a subset of PA that allows the chance constraint to be satisfied as OpSAT.

1. **Decision variables**: Binary $B_i \in \{\text{True,False}\}$ for each labeled constraints in PA;
2. **Objective function**: $\min \Pr(L_{B=\text{False}})$, where $L_{B=\text{False}}$ is the set of all labels of constraints such that B_i=False;
3. **Constraints**: $\Pr(L_{B=\text{False}}) \leq \Delta$ and $C_{B=\text{True}} \wedge NA$ is consistent, where $C_{B=\text{True}}$ is the set of all constraints such that B_i=True.

The Boolean variable B_i represents whether the labeled constraints $L_i \Rightarrow C_i$ in the constraint region *PA* is covered or not. In this formulation, a schedule extracted from a consistent set of temporal constraints $C_{B=\text{True}} \wedge NA$ has risk given by

$$\Pr(L_{B=\text{False}}) = \Pr\left(\bigcup_{i:B_i=\text{False}} L_i\right). \tag{4}$$

The probability (4) is computed using the Inclusion-Exclusion principle. However, since we are only concerned about the risk being below Δ, it is possible to compute simpler upper and lower bounds for $\Pr(L_{B=\text{False}})$ using the Bonferroni inequalities (Comtet 1974), as shown in Algorithm 2. A particular case of the Bonferroni inequalities is the upper bound $\Pr(L_{B=\text{False}}) \leq \sum_{i:B_i=False} \Pr(L_i)$, known as Boole's inequality.

Algorithm 2 Sequential probability approximations.

Input: Set of labels $L_{B=\text{False}}$, prob. model P, risk threshold Δ.
Output: Whether $\Pr(L_{B=\text{False}})$ is below, above, or is equal to Δ.

```
1: bound ← 0
2: for i = 1 to |L_{B=False}| do
3:     [e_1, ..., e_m] ← All subsets of L_{B=False} with i elements.
4:     probInc ← Σ_{i=1}^{m} Pr(e_i)
5:     if i is odd then bound ← bound + probInc
6:         if bound < Δ then return BELOW
7:     else bound ← bound − probInc
8:         if bound > Δ then return ABOVE
9: return EQUAL
```

The worst case performance of Algorithm 2 is equivalent to computing (4) and comparing it with Δ. The procedure for determining CCSC of pTPNs in Algorithm 3 is explained below.

Lines 2,3 *Creates an OpSAT instance for reward maximization.*

Lines 4,5 *Partitions constraints as described in this section. If NA is inconsistent, learns a new conflict and backtracks immediately.*

Lines 6-8 *Creates an OpSAT instance for risk minimization as described in this section. Risk bounds are computed with Algorithm 2. If the chance-constraint is violated, learns a new conflict and backtracks.*

Lines 9-12 *If the chance-constraint is satisfied with a consistent STN, returns the assignment $candCS$ that maximizes reward and the STN from which schedules can be extracted. Otherwise, learns a new conflict in terms of infeasible constraints.*

Algorithm 3 Chance-constrained strong consistency.

Input: pTPN, risk threshold Δ.
Output: Controllable scenario CS and feasible STN.

1: $LC \leftarrow toLC(pTPN)$
2: $OpS1 \leftarrow RewardMax(pTPN)$; $cConf \leftarrow \emptyset$
3: **while** $(candCS \leftarrow NextBest(cConf, OpS1)) \neq \emptyset$ **do**
4: $\{NA, NI, PA\} \leftarrow Partition(candCS, LC)$
5: **if** NA is inconsistent **then** BREAK
6: $OpS2 \leftarrow RiskMin(NA, PA, pTPN)$; $bConf \leftarrow \emptyset$
7: **while** $(candB \leftarrow NextBest(bConf, OpS2)) \neq \emptyset$ **do**
8: **if** $\Pr(L_{B=\text{False}}(candB)) > \Delta$ **then** BREAK
9: **if** $C_{B=\text{True}}(candB) \wedge NA$ is consistent **then**
10: **return** $\{candCS, C_{B=\text{True}}(candB) \wedge NA\}$
11: **else**
12: $bConf \leftarrow LearnConflict(bConf, candB)$
13: $cConf \leftarrow LearnConflict(cConf, candCS)$
14: **return** FAIL

5 Numerical chance-constraint evaluation

The results in this section only concern the OpSAT problem in lines 6-12 of Algorithm 3, which is responsible for the challenging problem of evaluating the chance-constraint. Figure 3 compares the relative average performance of our conflict-directed approach versus the current practice using chronological search (CS) when evaluating the chance-constraint for strongly consistent policies on a set of randomly generated pTPNs. Conflict-directed methods have the additional overhead of learning conflicts whenever infeasibility is detected, so it was not clear whether they would perform better when trying to evaluate the chance-constraint.

When dealing with hand-written examples, searching for a CCSC policy using CS or CDA* yielded the same performance. For small instances, there is no practical value on learning conflicts, since CS is able to explore all solutions in a matter of milliseconds. Thus, no useful conclusions could be drawn. Therefore, we randomly generated labeled temporal constraints conditioned on controllable and uncontrollable choices with moderately-sized domains (about 10 elements each). Probability models were also randomly generated. Simple temporal constraints were created between random pairs of nodes with varying bounds. The goal was to either find a CCSC schedule or return that no solution existed. Our implementation of CDA* searches for a strongly consistent policy as explained in Section 4.3, while CS tries

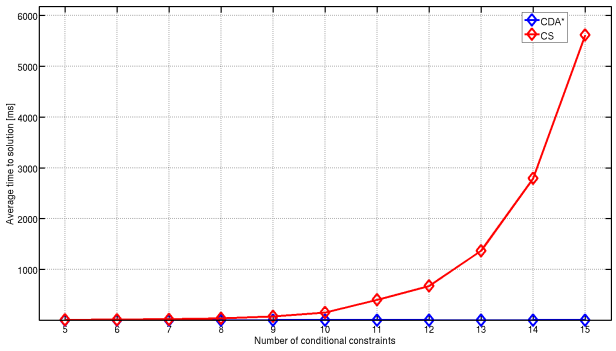

Figure 3: Average time to solution for CDA* *versus* CS.

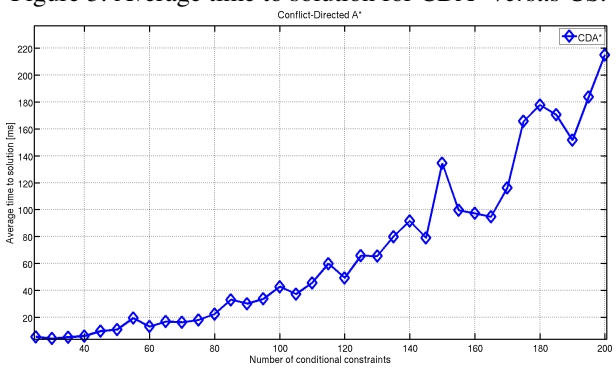

Figure 4: Average time complexity growth for CDA*.

to find a feasible STN by relaxing temporal constraints in best-first order of risk.

A problem with N conditional constraints induces a search space of size 2^N possible solutions. Both algorithms were run until the first candidate satisfying the risk bound Δ was found or no more solutions were available. Whenever more than one solution existed, CDA* returned the one incurring minimum risk. For relatively small plans with no more than 10 conditional constraints, we see that CS and CDA* showed very similar performances. However, if one increases the size of the problem by a few more constraints, we see a strong exponential growth in the time required by CS to either find a solution or return that no solution exists. Our approach using CDA*, on the other hand, kept its performance virtually unchanged. Despite the exponential trend in Figure 4 for CDA*, we see that it happens at a much smaller rate than for CS.

It is worthwhile to mention that CS was able to find feasible STNs quickly when the "hard" temporal constraints causing infeasibility had low probabilities assigned to them. In this cases, it was easy to restore feasibility of the STN by relaxing these low probability constraints while meeting the chance-constraint. It ran into troubles, however, whenever temporal constraints causing the STN to be infeasible were assigned high probabilities. In these cases, CS preferred to explore numerous lower risk alternatives before realizing that the source of infeasibility had high probability and, therefore, violated the chance-constraint. In these situations, the conflict extraction overhead paid off by quickly

revealing the problematic constraints with high probabilities and determining that the chance-constraint was infeasible.

6 Conclusions

This work introduced a representation of contingent plans with uncertainty, the Probabilistic Temporal Plan Network (pTPN), and presented formal definitions and algorithms for determining two novel types of consistency guarantees, namely chance-constrained weak and strong consistency. Our goal was to extend existing consistency guarantees for set-bounded representations of uncertainty to a more useful setting where execution policies are allowed to have an acceptable level of risk.

In the presentation of strong consistency, we introduced an efficient algorithm for evaluating the feasibility of the chance-constraint and generating a strongly consistent tightened schedule based on a conflict-directed "diagnostic" approach. This is an important result by itself, since it provides a way to quickly assess the risk of conditional plans with no contingencies with direct applications to the operation of remote agents in uncertain, hazardous environments.

Acknowledgments

Thanks to the anonymous reviewers and our colleagues Andrew Wang, Cheng Fang, Steven Levine, Peng Yu, and David Wang for their invaluable comments. We would also like to thank Mitch Ingham for sharing his experience with flight missions at JPL. This research is funded by AFOSR grant 6926079.

References

Birge, J. R., and Louveaux, F. V. 1997. *Introduction to stochastic programming*. Springer.

Comtet, L. 1974. *Advanced Combinatorics: The art of finite and infinite expansions*. Springer.

De Kleer, J. 1986. An assumption-based tms. *Artificial intelligence* 28(2):127–162.

Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D.; and Washington, R. 2003. Incremental contingency planning. In *ICAPS03 Workshop on Planning under Uncertainty and Incomplete Information*, 38–47.

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

Drummond, M.; Bresina, J.; and Swanson, K. 1994. Just-in-case scheduling. In *AAAI*, volume 94, 1098–1104.

Effinger, R.; Williams, B.; Kelly, G.; and Sheehy, M. 2009. Dynamic Controllability of Temporally-flexible Reactive Programs. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 09)*.

Effinger, R. T. 2006. Optimal Temporal Planning at Reactive Time Scales via Dynamic Backtracking Branch and Bound. Master's thesis, Massachusetts Institute of Technology.

Effinger, R. T. 2012. *Risk-minimizing program execution in robotic domains*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Fargier, H.; Lang, J.; Martin-Clouaire, R.; and Schiex, T. 1995. A constraint satisfaction framework for decision under uncertainty. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 167–174. Morgan Kaufmann Publishers Inc.

Fargier, H.; Lang, J.; and Schiex, T. 1996. Mixed constraint satisfaction: A framework for decision problems under incomplete knowledge. In *Proceedings of the National Conference on Artificial Intelligence*, 175–180.

Gelle, E., and Sabin, M. 2006. Solver framework for conditional constraint satisfaction problems. In *Proceeding of European Conference on Artificial Intelligence (ECAI-06) Workshop on Configuration*, 14–19.

Hunsberger, L.; Posenato, R.; and Combi, C. 2012. The Dynamic Controllability of Conditional STNs with Uncertainty. In *Proceedings of the Planning and Plan Execution for Real-World Systems: Principles and Practices (PlanEx) Workshop*, 121–128.

Kim, P.; Williams, B. C.; and Abramson, M. 2001. Executing reactive, model-based programs through graph-based temporal planning. In *IJCAI*, volume 17, 487–493.

Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *IJCAI*, volume 1, 494–502.

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

Tarim, S. A.; Manandhar, S.; and Walsh, T. 2006. Stochastic constraint programming: A scenario-based approach. *Constraints* 11(1):53–80.

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

Venable, K. B.; Volpato, M.; Peintner, B.; and Yorke-Smith, N. 2010. Weak and dynamic controllability of temporal problems with disjunctions and uncertainty. In *Workshop on Constraint Satisfaction Techniques for Planning & Scheduling*, 50–59.

Vidal, T., and Ghallab, M. 1996. Dealing with uncertain durations in temporal constraint networks dedicated to planning. In *ECAI*, 48–54.

Vidal, T. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence* 11(1):23–45.

Walcott, A. 2004. Unifying Model-Based Programming and Path Planning Through Optimal Search. Master's thesis, Massachusetts Institute of Technology.

Williams, B. C., and Ragno, R. J. 2007. Conflict-directed A* and its role in model-based embedded systems. *Discrete Applied Mathematics* 155(12):1562–1595.

Williams, B. C.; Ingham, M. D.; Chung, S. H.; and Elliott, P. H. 2003. Model-based programming of intelligent embedded systems and robotic space explorers. *Proceedings of the IEEE* 91(1):212–237.