Resolving Over-constrained Probabilistic Temporal Problems through Chance Constraint Relaxation

Peng Yu and Cheng Fang and Brian Williams
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA 02139
{yupeng,cfang,williams}@mit.edu

Abstract

When scheduling tasks for field-deployable systems, our solutions must be robust to the uncertainty inherent in the real world. Although human intuition is trusted to balance reward and risk, humans perform poorly in risk assessment at the scale and complexity of real world problems. In this paper, we present a decision aid system that helps human operators diagnose the source of risk and manage uncertainty in temporal problems. The core of the system is a conflict-directed relaxation algorithm, called Conflict-Directed Chance-constraint Relaxation (CDCR), which specializes in resolving over-constrained temporal problems with probabilistic durations and a chance constraint bounding the risk of failure. Given a temporal problem with uncertain duration, CDCR proposes execution strategies that operate at acceptable risk levels and pinpoints the source of risk. If no such strategy can be found that meets the chance constraint, it can help humans to repair the over-constrained problem by trading off between desirability of solution and acceptable risk levels. The decision aid has been incorporated in a mission advisory system for assisting oceanographers to schedule activities in deep-sea expeditions, and demonstrated its effectiveness in scenarios with realistic uncertainty.

Introduction

From transit times to weather changes, uncertainty exists in every real world scheduling problems. For example, in deep-sea expeditions, the unbounded uncertainty in the environment, the underwater vehicles and the crew performance make it impossible to find a plan that offers a 100% guarantee of success. Therefore, correct handling of uncertainties and management of risk are essential requirements for the ocean scientist who manages expedition plans. The scientist should use models of uncertainty to generate schedules that will operate successfully within specified risk bounds. When the problem is over-constrained, that is, no solution exists that satisfies all temporal constraints within the acceptable risk level, the scientist should make trade-offs between risk and performance to restore the feasibility of the mission.

Such a problem can be modeled by the chance-constrained probabilistic simple temporal problem (cc-pSTP) formulation, first presented in (Fang, Yu, and Williams 2014). A solution to a cc-pSTP is a strategy for executing its activities such that the chance of violating any temporal constraints is lower than the chance constraint. If, however, no feasible solution exists for the problem, we would like to pinpoint sources of risk, and to explore resolutions to the problem that trade off risk with performance. Acceptable risk levels in a mission may be negotiable: in some urgent situations, the scientists are willing to make compromises by taking more risk in order to complete a top priority task. This motivates us to develop a decision aid that can automatically identify the causes of failure and recommend good resolutions in such over-constrained situations. It works like an experienced mission planner and helps the scientists tackle problems of larger scale and complexity, find better solutions, and speed up the mission planning process.

In this paper, we present the key algorithm of the decision aid, Conflict-Directed Chance-constraint Relaxation (CDCR), which computes preferred trade-offs between temporal and chance constraint relaxations for over-constrained cc-pSTPs. In the literature, several methods have been introduced for scheduling cc-pSTPs (Fang, Yu, and Williams 2014; Tsamardinos 2002), or resolving over-constrained temporal problems (STPs (Beaumont et al. 2001; 2004), DTPs (Moffitt and Pollack 2005), CCTPs (Yu and Williams 2013) and STNUs (Yu, Fang, and Williams 2014)). Our approach leverages prior work on both probabilistic scheduling and temporal relaxation. The key idea is to diagnose the source of risks by grounding pSTPs to STNUs through risk allocation over probabilistic uncertain durations, then apply controllability checking and conflict extraction algorithms to identify conflicting constraints from the grounded problem. Resolutions to these conflicts can then guide us to find (1) adjustments for the risk allocation; (2) relaxations for temporal constraints; or (3) relaxations for the chance constraint. This resolution process is iterative: CDCR continues to discover and resolve new conflicts, until (a) a feasible grounded STNU is found, which indicates that the current resolution repairs all conflicts in the problem; or (b) no resolution can be found for all known conflicts, which indicates that the cc-pSTP cannot be resolved.

For example, a scientist is planning to deploy an underwater robot to survey a volcano eruption on the sea floor. The eruption will occur at around 10:00, following a normal

distribution with a variance of 30 minutes. It is 8:00 now, and the robot needs to arrive at the site before the start of the eruption. In addition, at least 45 minutes is required for traversing to the site, and 30 minutes for collecting samples. The scientist wants the mission to complete in 3 hours, with less than 5% risk of violating any constraints, such as being late or missing the event. We can capture this problem using a cc-pSTP (Figure 1). After evaluating all the requirements, the decision aid (DA) determines that no solution exists that meets all requirements. It immediately engages the scientist (ST) and initiates a discussion to resolve this problem.

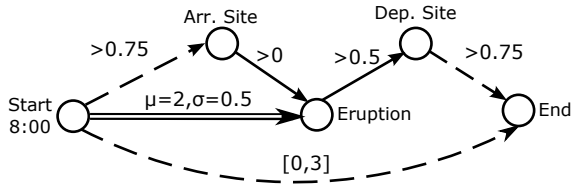

Figure 1: The cc-pSTP for the underwater robot's mission

DA: *I cannot meet all requirements due to the limited mission time and the uncertainty in eruption. Can you* ***extend the mission*** *to 4 hours and 10 minutes.*

ST: *You can have at most 4 hours for this mission.*

DA: *May I* ***increase the risk bound*** *for this mission to 7.3% in order to meet the duration requirement?*

ST: *I do not want to take that much risk on this task.*

DA: *Ok, can you* ***shorten the traversal time*** *from the site to the ship by 6 minutes? My plan can then cover 95% of the possible eruption time, between 8:45 and 10:51.*

ST: *That's fine. Thanks.*

This example demonstrates the desired features of CDCR: diagnosing over over-constrained cc-pSTPs and compute resolutions through trade-offs between relaxations over chance and temporal constraints. More importantly, it works collaboratively with the users during the process, proposing alternatives and learning new requirements, in order to find better resolutions for them. CDCR has been incorporated as part of a mission advisory system for helping oceanographers manage their expeditions. It assists in the planning for underwater vehicle activities while maintaining an acceptable risk level and maximizing the missions' scientific return. In the following sections, we present the design, implementation and empirical results of the CDCR algorithm.

Problem Statement

The CDCR algorithm takes cc-pSTPs as inputs. cc-pSTP is an extension to the Simple Temporal Problem formulation (STP (Dechter, Meiri, and Pearl 1991)). In addition to the simple temporal constraints in STPs, it adds two new types of constraints to the problem: probabilistic temporal constraints for modeling uncertain durations, and chance constraint for specifying the acceptable level of risk. Compared to the set-bounded contingent constraints used in the Simple Temporal Networks with Uncertainty formulation (STNUs (Vidal and Fargier 1999)), the probabilistic representation of uncertain durations allows cc-pSTP to more accurately model uncertainty in real world activities. In addition, the chance constraint allows a quantified bound on risk taken to be specified, which is more flexible and intuitive than the criteria of controllability. Here, we repeat the definition of cc-pSTP from (Fang, Yu, and Williams 2014) for reference.

Definition 1. *A cc-pSTP is a pair* $\langle N^+, \Delta_t \rangle$*, where:*

- N^+ *is a probabilistic Simple Temporal Network (pSTN), defined as a 4-tuple* $\langle X_b, X_e, R_c, R_d \rangle$*, where:*
  - X_b *is a set of controllable events whose times can be assigned by the agent.*
  - X_e *is a set of uncontrollable events whose times are assigned by the external world.*
  - R_c *is a set of requirement constraints between controllable events. Each* $c_{xy} \in R_c$ *is of type* $(y - x) \in [l_{xy}, u_{xy}]$*, where* $x, y \in X_b \cup X_e$*.*
  - R_d *is a set of probabilistic uncertain durations. Each* $d_{xy} \in R_d$ *is a random variable describing the difference* $(y - x)$*, where* $x \in X_b$ *and* $y \in X_e$*.*
- $\Delta_t \in [0, 1]$ *is the chance constraint that sets the upper bound on the risk of failure, for the set of requirement constraints* R_c *in* N^+*.*

Definition 2. *A cc-pSTP solution is a pair* $\langle N_g, E_x \rangle$*, where:*

- N_g *is a grounded STNU of the cc-pSTP. It specifies the allocated risk over each probabilistic duration: the lower and upper bounds allocated for each duration indicate the range of outcomes covered. The total amount of uncovered outcomes across all probabilistic durations must be smaller than the chance constraint.*
- E_x *is an execution strategy for* N_g*. It covers all controllable events in the cc-pSTP, and is controllable with regards to* N_g*.*

Given a cc-pSTP P with chance constraint Δ_t, if we execute the controllable events using E_x in its solution, the chance of violating any temporal constraints in P is guaranteed to be less than Δ_t. The policy E_x could be a static schedule (with a strongly controllable N_g), or a dynamic execution policy (with a dynamically controllable N_g). In this paper, we attempt to find dynamic execution policies to allow for flexibility in responding to the outcomes of uncertain durations.

If a cc-pSTP is over-constrained, no solution exists that can meet all temporal constraints within the risk bound. In other words, there is no N_g that is controllable and takes less risk than the chance constraint. This occurs when the user specifications are too restrictive, for example when the desired time bounds are too tight, or when the user is overly cautious in setting the chance constraint. These problems can be resolved through constraint relaxations. We thus define a relaxable version of the cc-pSTP formulation that allows some of its temporal and chance constraints to be relaxed at a cost.

Definition 3. *A relaxable cc-pSTP contains all elements from a cc-pSTP plus four additional elements,* rR_c*,* f_{rc}*,* $R\Delta_t$ *and* f_Δ*, where:*

- *rR_c is a set of requirement constraints whose temporal bounds can be relaxed, $rR_c \subseteq R_c$.*
- *$f_{rc} : (c_{xy}, c'_{xy}) \rightarrow \mathbb{R}^+$ is a function that maps the relaxation to a relaxable constraint, $c_{xy} \rightarrow c'_{xy}$ where $c_{xy} \in rR_c$, to a positive cost value.*
- *$r\Delta_t \in [T, F]$ is a boolean value that indicates if the chance constraint can be relaxed to a higher value.*
- *$f_\Delta : (\Delta_t, \Delta'_t) \rightarrow \mathbb{R}^+$ is a function that maps the relaxation to the chance constraint, $\Delta_t \rightarrow \Delta'_t$ where $\Delta_t \leq \Delta'_t \leq 1$, to a positive cost value.*

Definition 4. *A valid resolution for an over-constrained cc-pSTP, P, is a 3-tuple $\langle R'_c, \Delta'_t, N_{alloc} \rangle$, where:*

- *R'_c is a set of relaxations (in terms of relaxed lower and upper bounds) to constraints in rR_c of P.*
- *Δ'_t is a relaxation for the chance constraint Δ_t of P, and $\Delta'_t \geq \Delta_t$.*
- *N_{alloc} is a STNU generated from P by grounding all probabilistic durations with fixed lower and upper bounds.*

such that N_{alloc} is dynamically controllable and covers more than $1 - \Delta'_t$ of the uncertain durations' outcomes.

Note that there are usually more than one valid resolutions to a cc-pSTP due to the continuous property of temporal and chance constraint relaxations. It is important to prioritize the resolutions and enumerate only preferred ones of lower costs for the users. In addition, the users may reject a resolution proposed by the decision aid, and ask for an alternative one with additional requirements. These newly added requirements should be respected by all future resolutions. Finding a good resolution usually requires a considerable amount of negotiation since the users may not have encoded all their constraints in the input problem. The decision aid needs to learn about them through the interaction before reaching an agreement with the user.

Approach

In this section, we present the design and implementation of the CDCR algorithm that resolves over-constrained cc-pSTPs. CDCR leverages ideas and methods from probabilistic scheduling and relaxation algorithms in the literature: it leverages the ideas from (Fang, Yu, and Williams 2014) for grounding probabilistic pSTPs into deterministic STNUs, and uses the conflict-directed framework from (Yu, Fang, and Williams 2014) for efficient conflict detection and resolution. We start with an overview of CDCR, then discuss the three major procedures of the algorithm: conflict detection, risk allocation and constraint relaxation.

Conflict-Directed Chance-constraint Relaxation

The input to the CDCR algorithm is a relaxable cc-pSTP. Given such a problem, CDCR enumerates feasible resolutions in best-first order: a resolution is a collection of relaxations for temporal and chance constraints, and each resolution supports a grounded STNU whose risk is bounded by the relaxed chance constraint. CDCR (Algorithm 1) takes a conflict-directed relaxation and allocation approach: given the grounded STNU of a cc-pSTP that represents a specific risk allocation, the algorithm identifies conflicts between constraints and uses their resolutions to guide the search towards feasible risk allocation and constraint relaxations. This is similar to the framework used by the CDRU algorithm (Yu, Fang, and Williams 2014), with two key modifications.

- First, an additional step of risk-allocation is required for grounding the probabilistic input problem to a STNU. This allows us to check the feasibility and extract conflicts between constraints using existing STNU algorithms.
- Second, in addition to temporal constraint relaxations, the conflict resolution step also adjusts risk allocation and adds necessary relaxations in order to resolve all known conflicts while maintaining the risk-taken below the (relaxed) chance constraint. Note that this step may require a non-linear optimization solver if the probabilistic distribution of any uncertain duration is non-linear.

Input: A cc-pSTP $\langle N^+, \Delta_t, rR_c, f_{rc}, r\Delta_t, f_\Delta \rangle$.
Output: A relaxation $\langle R'_c, \Delta'_t, N_{alloc} \rangle$ that minimizes $f_{rc} + f_\Delta$.
Initialization:
1 $Cand \leftarrow \langle R_{cand}, \Delta_{cand}, N_{cand}, C_r \rangle$; the first candidate;
2 $Q \leftarrow \{Cand\}$; a priority queue of candidates;
3 $C \leftarrow \{\}$; the set of all known conflicts;
Algorithm:
4 **while** $Q \neq \emptyset$ **do**
5 $Cand \leftarrow$Dequeue(Q);
6 $currCft \leftarrow$UNRESOLVEDCONFLICT($Cand, C$);
7 **if** $currCft == null$ **then**
8 $newCft \leftarrow$DYNAMICCONTROLLABLE?($Cand$);
9 **if** $newCft == null$ **then**
10 return $Cand$;
11 **else**
12 $C \leftarrow C \cup \{newCft\}$;
13 $Q \leftarrow Q \cup \{Cand\}$;
14 **endif**
15 **else**
16 $Q \leftarrow Q \cup$EXPANDONCONFLICT$\{Cand, currCft\}$;
17 **endif**
18 **end**
19 return $null$;

Algorithm 1: The CDCR algorithm

CDCR is implemented with a priority queue for enumerating resolutions in best-first order. The algorithm starts with an empty candidate (Line 1) that has no relaxations over temporal constraints (R_{cand}) and chance constraint (Δ_{cand}), and an empty set of resolved conflicts (C_r). The candidate is associated with a risk allocation over all probabilistic uncertain durations, which is represented by a STNU (N_{cand}). The initial allocation is computed from a non-linear solver and is conservative enough to meet the chance constraint. The initial candidate is the only element in the queue before search starts (Line 2).

Within the main loop, CDCR first dequeues the best candidate (Line 5) and checks if it resolves all known conflicts (Line 6). If not, a conflict $currCft$ will be returned by function UNRESOLVEDCONFLICT. The unresolved con-

flict is then used for expanding $Cand$ (Line 16). All child candidates returned by function EXPANDONCONFLICT resolve $currCft$, and are added back to the queue for future evaluation and expansion.

If $Cand$ resolves all known conflict, CDCR will proceed to check the controllability of its grounded STNU (function DYNAMICCONTROLLABLE?, Line 8). If controllable, $Cand$ will be returned as the best resolution to the cc-pSTP (Line 10). Otherwise, a new conflict will be returned by this function and recorded for expanding candidates (Line 12). $Cand$ will also be added back to Q since it now has an unresolved conflict (Line 13).

Conflict Learning from Controllability Checking

Conflict learning is the key for resolving over-constrained cc-pSTPs. Conflicts explain the cause of failure and provide guidance for computing necessary relaxations. A conflict is represented by a conjunctive set of linear expressions. Each expression is a necessary constituent of the conflict, and is defined over the lower and upper bounds of some requirement and contingent constraints, with integer coefficients. CDCR learns new conflicts iteratively from grounded STNUs with different risk allocations. The method for learning conflict from controllability checking algorithms was introduced in (Yu, Fang, and Williams 2014). Here we present a brief review of the method using a simple dynamic controllability example (Figure 2).

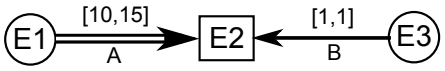

Figure 2: The original STNU

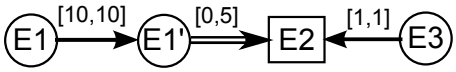

Figure 3: The normalized STNU

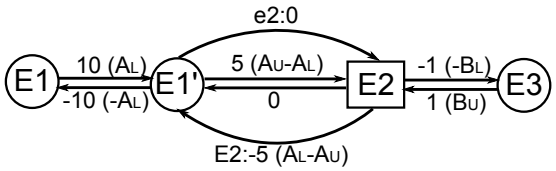

Figure 4: The equivalent distance graph of the STNU

There are three events, E1, E2 and E3, in this example STNU. These events are connected by two constraints A and B: A is uncontrollable with a bound of [10,15], while B is controllable with a bound of [1,1]. The first step of controllability checking is to map the STNU to a normalized form (Morris and Muscettola 2005), which decouples the lower bounds from each uncontrollable duration (Figure 3). We can then generate the equivalent distance graph using the normalized STNU. Note that each distance edge in the graph, including conditional edges, is labeled with a linear expression over constraints. The expression represents the source of its weight value (Figure 4).

Next, we identify and reduce all moat paths in the distance graph using the iterative method introduced in (Morris 2006). In this example, there is only one valid moat path: $E1' \rightarrow E2 \rightarrow E3$. This path has a negative weight, starts with a lower-case edge, and can be reduced to a single edge using a lower-case reduction. The reduced edge (represented by a dotted arrow in Figure 5) of the moat path has a weight of -1, and is supported by a linear expression that combines the expressions of all edges in the moat path.

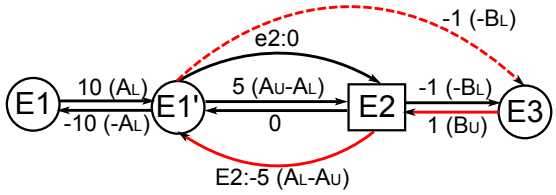

Figure 5: The distance graph with a reduced edge

After all applicable reductions, the final step is to run all max consistency check on the graph, which excludes all lower-case edges. It reveals any negative cycle in the reduced graph, whose existence indicates that the STNU is not dynamically controllable. In this example, one negative cycle can be detected that contains edge $E3 \rightarrow E2$, $E2 \rightarrow E1'$, and the reduced edge $E1' \rightarrow E3$. From this cycle, we can identify the linear expression that caused this conflict from all distance edges in the cycle: $B_U + A_L - A_U - B_L \leq 0$. In addition, there is another subtle but necessary element of this conflict: the reduction that adds edge $E1' \rightarrow E3$. The negative cycle would not exist without this reduced edge. Therefore, the expression that supports the reduction, $-B_L \leq 0$, which guarantees a negative weight for the moat path, is included in the conflict. The conflict we can extract from the STNU is a conjunction of two linear expressions: $B_U + A_L - A_U - B_L \leq 0$ and $-B_L \leq 0$.

In summary, learning conflicts from STNUs requires recording the supporting expression for each distance edge and reduction. Once a negative cycle is detected, we can extract a conflict by collecting (1) the expressions for each edge in the cycle; and (2) the expressions required by the reductions that added edges to the cycle. The conflict is a conjunction of these linear expressions, which are all negative and defined over the temporal bounds of constraints.

Risk Allocation and Constraint Relaxation

Conflicts provide guidance for CDCR to resolve over-constrained problems. A conflict may be resolved in multiple ways: it is eliminated if any of its linear expressions is made non-negative. For example, we can resolve the conflict in Figure 5 using the following two approaches:

- Set $B_U + A_L - A_U - B_L \geq 0$, e.g. increasing A_L to 15.
- Set $-B_L \geq 0$, e.g. lowering B_L to 0.

Intuitively, to resolve a conflict we can directly require the weight of a previously negative cycle to be non-negative, or we can make sure the reduction which adds an edge never occurs. This choice in resolution is unique to dynamic controllability conflicts: a conflict from consistency or strong controllability checking only has one linear expression. It provides more flexibility in conflict resolution, although it also increases the complexity of the problem: to enumerate resolutions in best-first order, CDCR may need to evaluate all possible repairs for all conflicts. The search branches each time CDCR expands on a conflict. If a quick response is desired by the user, CDCR can be implemented with anytime search strategies.

Given a set of conflicts, we can formulate a constrained optimization problem and compute the resolutions using a non-linear optimization solver. There are three categories of variables in the optimization problem: relaxations for temporal constraints (rc'_{iL} and rc'_{iU}), relaxations for chance constraint (Δ'_t), and the allocation of lower and upper bounds for probabilistic durations (rd'_{iL} and rd'_{iU}). Each category of variables represents a type of conflict resolutions: re-allocating risk over probabilistic durations, relaxing chance constraint, and relaxing temporal constraints. These are given in Problem 1.

Problem 1 (Conflict resolution)**.**

$$\min_{\Delta'_t, rc_j \in rR_c} f_\Delta(\Delta'_t - \Delta_t) + \sum_{j=1}^{|rR_c|} f_{rc}(rc_j) \tag{1}$$

$$s.t. \quad rd'_{iL} - rd'_{iU} < 0 \tag{2}$$

$$rc'_{jL} - rc_{jL} \leq 0, \qquad rc'_{jU} - rc_{jU} \geq 0 \tag{3}$$

$$rc'_{kL} - rc_{kL} = 0, \qquad rc'_{kU} - rc_{kU} = 0 \tag{4}$$

$$Conflict_m \geq 0 \tag{5}$$

$$\sum_{rd \in R_d} \textsc{Risk}(rd_{iL}, rd_{iU}) \leq \Delta'_t,\ \Delta'_t \in [\Delta_t, 1) \tag{6}$$

The constraints in the optimization problem enforce the necessary properties. For lower and upper bound variables of probabilistic durations, their value can be freely assigned as long as the lower bound is smaller than the upper bound, encoded by (2). For relaxable requirement constraints, their new temporal bounds must be no tighter than the original bounds, as in (3). For requirement constraints that are not relaxable, their temporal bounds remain unchanged as in (4).

The resolution constraints in (5) are added to ensure that all known conflicts are repaired by the resolution. Given m conflicts, the same number of resolution constraints will be added, each representing the negation of one linear expression in each conflict. For example, given two conflicts $A_L - 5B_U + 2C_L < 0$ and $2B_L + C_L + D_U < 0$, the resolution constraints will be $A_L - 5B_U + 2C_L \geq 0$ and $2B_L + C_L + D_U \geq 0$. Variables covered by these constraints are the lower and upper bounds of both requirement constraints and probabilistic durations, with integer coefficients.

Finally, we add risk allocation constraint to ensure that the risk taken meets the chance constraint. This constraint is defined over the lower and upper bound variables of all probabilistic durations. Given distributions of each probabilistic duration and the uncertainty bounds chosen, the RISK function computes the probability mass of the regions outside the uncertainty bounds. CDCR uses the union bound to upper-bound the total risk taken across all uncertain durations, as this does not rely on assumptions of independence. If the chance constraint is relaxable, we further require that the relaxed chance constraint is lower-bounded by the original chance constraint and 1. This gives us the flexibility to make trade-offs between risk and performance, if no solution can be found that resolves all conflicts while meeting the current chance constraint. These are described by (6).

The objective function, given in (1), is defined over f_Δ and f_{rc} for minimizing the cost of temporal and chance constraint relaxations. In the optimization problem, all domain and conflict resolution constraints are linear, while the chance constraint may be non-linear depending on the probabilistic distributions. CDCR uses the Interior Point Optimizer (IPOPT (Wchter and Biegler 2006)) to compute optimal resolutions. If a resolution is returned by IPOPT, function EXPANDONCONFLICT will construct a new candidate with its constraint relaxations and the risk allocations. This candidate will then be added as a new branch to CDCR's search tree.

Incorporating User Inputs

Finally, we present the implementation of CDCR that takes user responses to improve existing solutions. This feature allows CDCR to accept new requirements from the user during the search, and incorporate them in all future solutions. It is achieved through adding a step to the algorithm presented in preceding section (Algorithm 2): if the user rejects the current solution, CDCR records their inputs as a new conflict and add it to the known conflicts list C (Line 11). Allowed responses are adjustments to both temporal constraints and risk allocations, such as "This mission must complete in 10 hours instead of 15", or "Task A is not as important, increase risk allowance from 5% to 10%".

Initialization:
1 $Sol \leftarrow \langle R'_c, \Delta'_t, N_{alloc} \rangle$; a solution to P;
2 $C \leftarrow \{\}$; the set of all known conflicts;
Algorithm:
3 **while** $true$ **do**
4 $\quad Sol \leftarrow$CDCR(C);
5 $\quad$ **if** $Sol == null$ **then**
6 $\quad\quad$ `return` $null$;
7 $\quad$ **else**
8 $\quad\quad$ **if** *Accepted?*(Sol) **then**
9 $\quad\quad\quad$ `return` Sol;
10 $\quad\quad$ **else**
11 $\quad\quad\quad$ $C \leftarrow C \cup$PARSEINPUTS$\{UserResponse\}$
12 $\quad\quad$ **endif**
13 $\quad$ **endif**
14 **end**

Algorithm 2: Addition to CDCR for Incorporating Inputs

Once an input has been recorded and inserted into the known conflict set, it will be used to expand future candidates. This guarantees that all future solutions respect this newly added requirement. CDCR also adds the current solution back to the queue, as it does not resolve the new conflict. It then restarts the search (Line 4) for solution that resolves all conflicts while minimizing cost.

Application and Experimental Results

The CDCR algorithm has been incorporated as part of a mission advisory system for helping oceanographers schedule tasks in deep-sea expeditions. Their missions are usually weeks long, and involve the operations of several underwater robots. Each robot usually performs ten to fifteen dives in a mission, and a dive may last anytime between 6 to 16 hours. During each dive, a robot is tasked with a set of survey locations on the sea floor: the robot needs to traverse to each location, take samples and images, and return before running out of power. Due to unexpected ocean currents and incomplete terrain data, the traversal time between survey sites is highly uncertain and difficult to estimate. It is almost impossible for the oceanographers to correctly assess the uncertainty of each dive and plan tasks accordingly to meet the risk bound and a large set of operational constraints.

The advisory system simplifies the oceanographers' tasks and significantly reduced their workload in these missions. The advisor can check the feasibility of a mission plan and search for valid risk allocations that meet the risk requirement. If no such allocation can be found, the decision aid will explain the cause of failure using the conflicts detected during the search, and propose preferred relaxations to resolve the over-constrained plan. If the users are not satisfied with the results, they can ask the mission advisor to adjust the solutions given their new inputs.

In the rest of the section, we present experimental results that demonstrate the run-time performance of CDCR on problems with different size and complexity. The test cases were generated using a mission simulator for underwater expeditions. Given a set of target locations on a map, the simulator generates survey tasks around them and connects these locations with transit activities. Each test case describes a dive of multiple survey tasks: the traversals are represented by probabilistic durations, while the survey times and battery restrictions are modeled by simple temporal constraints. The operational risk limit is specified by the chance constraints in the cc-pSTPs. In addition, multiple underwater robots may be deployed and working in parallel during a dive: each robot may have different speed and power storage. Depending on the distance and vehicles, probabilistic durations of different traversals and robots may have different distributions.

In total, we created 2000 test cases using randomly generated numbers of vehicles, risk bounds, task locations and mission length. For each problem, we run CDCR to find a grounded STNU of the cc-pSTP that is dynamically controllable while meeting the chance constraint, or a set of relaxations for the cc-pSTP that will enable such a STNU. The timeout for each test is 30 seconds, which is usually the maximum duration that users are willing to wait for.

Results

We benchmarked CDCR on each problem twice: the first run assumes normal distribution over the uncertain durations, and the second run assumes uniform distribution. The cutoff for the uniform distribution is selected using $\mu \pm 2\sigma$. In each test, the time consumption of the first solution returned was recorded. The results are shown in Figure 6.

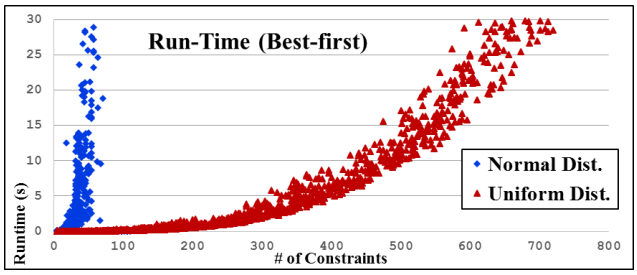

Figure 6: Runtime of CDCR (first solution)

In total, 453 of 2000 normal distribution tests were solved by CDCR in 30 seconds, while the number for uniform distribution tests is 1640 of 2000. As can be seen from the figure, CDCR performs much better on problems with uniform distribution, since all constraints are linear during conflict resolution in uniform distribution tests (we used the union bound approximation). This requires significantly less computation compared to normal distribution tests. Solving the optimization problems for conflict resolution is the most expensive operation in the CDCR algorithm, which takes up to 90% of the total computation time. Problems with normal distribution durations add a highly non-linear chance constraint during conflict resolution, making it much harder for the optimizer to solve.

On the other hand, normal distribution is a better model for the uncertainty in some real world activities, for example the timing of natural phenomena. Even though it requires much more computation than uniform distribution problems of the same size, CDCR resolved most of the problems with less than 100 constraints in 30 seconds. This is enough for modeling a 10-hour survey mission of several robots working in parallel. In general, the algorithm configuration and modeling should be different from one application to another. The choice of distributions to use is thus application dependent, although these results suggest that tractability must be considered when selecting density functions.

Contribution

In this paper, we presented the Conflict-Directed Chance-constraint Relaxation algorithm, the first approach for resolving over-constrained probabilistic temporal problems with chance constraints. CDCR leverages prior work on probabilistic scheduling and temporal relaxation to diagnose the source of risks and enumerate preferred resolutions with bounded risk. It allows the users to resolve conflicts by trading off between safety margins and performance, providing flexibility for risk management in real world scenarios. CDCR has been incorporated in a mission advisory system

for assisting oceanographers to plan and schedule activities in deep-sea expeditions, with input from the user. We have numerically demonstrated that the computational time required by CDCR is tractable for a class of underwater survey missions. The CDCR algorithm thus provides a principled decision-making assistance for scheduling under uncertainty, working with human operators to produce policies with probabilistic guarantees on temporal consistency.

Acknowledgments

Thanks to Eric Timmons, David Wang, Richard Camilli and Scott Smith for their support. This project is funded by the Boeing Company under grant MIT-BA-GTA-1.

References

Beaumont, M.; Sattar, A.; Maher, M.; and Thornton, J. 2001. Solving overconstrained temporal reasoning problems. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AI-2001)*, 37–49.

Beaumont, M.; Thornton, J.; Sattar, A.; and Maher, M. 2004. Solving over-constrained temporal reasoning problems using local search. In *Proceedings of the 8th Pacific Rim Conference on Artificial Intelligence (PRICAI-2004)*, 134–143.

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

Fang, C.; Yu, P.; and Williams, B. 2014. Chance-constrained probabilistic simple temporal problems. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, 2264–2270.

Moffitt, M. D., and Pollack, M. E. 2005. Partial constraint satisfaction of disjunctive temporal problems. In *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS-2005)*, 715–720.

Morris, P., and Muscettola, N. 2005. Temporal dynamic controllability revisited. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-2005)*, 1193–1198.

Morris, P. 2006. A structural characterization of temporal dynamic controllability. In *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP-2006)*, 375–389.

Tsamardinos, I. 2002. A probabilistic approach to robust execution of temporal plans with uncertainty. In *Methods and Applications of Artificial Intelligence: Proceedings of the Second Hellenic Conference on Artificial Intelligence*, volume 2308 of *Lecture Notes in Computer Science*, 97–108.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence* 11:23–45.

Wchter, A., and Biegler, L. T. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106:25–57.

Yu, P., and Williams, B. 2013. Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI-2013)*, 2429–2436.

Yu, P.; Fang, C.; and Williams, B. 2014. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-2014)*, 341–349.