Towards Temporally Uncertain Explainable AI Planning

Andrew Murray[1], Benjamin Krarup[2], and Michael Cashmore[1]

[1] Strathclyde University, Glasgow, UK.
{michael.cashmore, a.murray}@strath.ac.uk
[2] King's College London, London, UK.
benjamin.krarup@kcl.ac.uk

Abstract. Automated planning is able to handle increasingly complex applications, but can produce unsatisfactory results when the goal and metric provided in its model does not match the actual expectation and preference of those using the tool. This can be ameliorated by including methods for explainable planning (XAIP), to reveal the reasons for the automated planner's decisions and to provide more in-depth interaction with the planner. In this paper we describe at a high-level two recent pieces of work in XAIP. First, plan exploration through model restriction, in which contrastive questions are used to build a tree of solutions to a planning problem. Through a dialogue with the system the user better understands the underlying problem and the choices made by the automated planner. Second, strong controllability analysis of probabilistic temporal networks through solving a joint chance constrained optimisation problem. The result of the analysis is a Pareto optimal front that illustrates the trade-offs between costs and risk for a given plan. We also present a short discussion on the limitations of these methods and how they might be usefully combined.

1 Introduction

Automated Planning is the process of considering and organising actions to achieve goals before starting to execute them. In automated planning, the actions that must be performed are not predetermined by the goals, but are selected and scheduled from a typically large number of alternative actions. The choice is guided by an effort to achieve the goals whilst optimising various metrics. Ordering choices and resource allocations are made, and evaluated, as part of the selection process. The consequence of this approach is that neither the number of actions in a plan, nor resource allocation of the plan, are predetermined. This distinguishes planning from scheduling, where the actions to be performed are predetermined but the timing of actions, and the allocation of resources to them, are not [16].

As automated planning is being used in increasingly complex applications, explanation plays an crucial role in building trust – both in automated planners and in the plans that they produce. A plan is a set of instructions that can can be carried out by humans or autonomous agents. In either case, the plan conveys the means by which a goal is to be achieved, but not the reasons for the choices it embodies. When the audience for a plan includes humans then it is natural to suppose that some users might wish to question the reasoning, intention and underlying assumptions that lead to those choices.

As a result of this, there has been growing interest in investigating the explanation of plans [3], developing various approaches to building trust and understanding in the decisions made by an automated planner. Automated planning presents a distinct advantage in the context of explainable AI (XAI) in that it relies on the use of a model of the available actions. The model supports both prediction of action effects on a state and the identification of states from which the actions are applicable. However, in the context of explanation, the model becomes a shared vocabulary between the system and the user, enabling a depth and specificity of communication that is more difficult to obtain when explaining the decisions of systems such as deep-neural nets [31].

In this paper we describe recent work in two distinct areas of explainable automated planning (XAIP). First, in Section 2 we report on recent work in plan negotiation and the use of *constrastive explanations* to explore a space of plans. Then, in Section 3 we report on an approach to optimisation under one kind of uncertainty: *temporal uncertainty* about how long actions will take to complete. This approach can be used to generate analyses of the trade-offs between plan costs and risks. In Section 4 we discuss how this approach might be embedded within iterative plan exploration.

Both of these approaches share the idea of exploring a space of solutions, and by so doing gain a deeper understanding about the structure of the underlying problem. The motivation behind these approaches is that the user, with increased understanding, will either (i) gain trust in the decisions made by the automated planner, (ii) identify where the planner is operating outside of its competency, or (iii) identify where the planner is unaware of the user's preferences. In whichever case, the process of automated planning becomes a more useful interactive process that has the potential to converge towards a more satisfactory plan. In Section 4 we discuss how these different strands could be brought together to form a comprehensive suite of tools for plan explanation, which could be used by a human operator to better understand the problem, constraints, and converge to a more preferred solution.

1.1 Explainable AI Planning in Literature

Meuller et al. [26] provide an overview of the landscape of research into XAI. This work spans several decades, and includes work carried out with intelligent tutoring systems, XAI hypotheses and models, and explanation in expert systems. The early work on explanation in expert systems provided causal explanation for conclusions, often in the form of chains of rules contributing to the conclusion [38]. Recently, there has been a resurgence of interest in explanation in XAI, both when the model is and is not interpretable. This is, in large part, due to the difficulty of understanding the results of deep learning systems [31].

While there is a long history of work on explanation in AI, most work on explanation of plans (XAIP) is relatively recent. In a challenge paper, Smith [33] argues for the importance of plan explanation in mission planning, and suggests that questioning and explanation is part of an iterative process that helps elucidate and refine the preferences for a planning problem. Fox et al. [14] highlight contrastive 'why' questions as being important for plan explanation, and describe a number of different types of these questions and possible responses. Chakraborti et al. [3] survey recent work in XAIP and categorise the different approaches that have emerged in the last several years, including three important areas:

– *model reconciliation* - namely, that the need for explanation is due to differences between the agent's and the human's model of the planning problem. The planning system therefore "suggests changes to the human's model, so as to make its plan be optimal with respect to that changed human model" [23, 2, 34].
– *contrastive explanations* - an approach answering local contrastive questions; explaining the reason that a contrast case *B* was not a feature of the plan by revealing the consequences that would hold if *B were* the case [7, 20, 19, 1, 14, 22].
– and *explanation of unsolvability* for planning problems [9, 8, 35, 17].

2 Contrastive Explanations in Plan Exploration

Fundamentally, the need for plan explanation is driven by the fact that a human and a planning agent may have different models of the planning problem and different computational capabilities. In this section we describe our approach to plan explanation through *exploration*. Through asking contrastive questions, a human user can impose iterative restrictions upon the model of the automated planner in order to generate different plans. In so doing, the user gains a better understanding of the problem and capabilities of the automated planner. An in-depth description of this process, including a formal definition of model restrictions, is presented in Krarup et al. [21].

2.1 Planning Model and Capability

A standard modelling language for autonomous planning is the Planning Domain Description Language (PDDL), originally developed in 1998 by a committee led by Drew McDermott [25] and later extended to support more expressive features such as time [13, 6], preferences [15], continuous change and exogenous events [12]. To describe our approach to plan exploration, we'll use PDDL2.1 as an example planning formalism. Our definition follows the definition of PDDL2.1 given by Fox & Long [13], extended by a set of time windows and explicit record of the plan metric. A more detailed description can be found in Krarup et al. [21].

Definition 1 (Planning model). *A planning model is a pair $\Pi = \langle D, Prob \rangle$. The domain $D = \langle Ps, Vs, As, arity \rangle$ is a tuple where Ps is a finite set of predicate symbols, Vs is a finite set of function symbols, As is a set of action schemas, called operators, and $arity$ is a function mapping all of these symbols to their respective arity. The problem $Prob = \langle Os, I, G, M, W \rangle$ is a tuple where Os is the set of objects in the planning instance, I is the initial state, G is the goal condition, M is a plan-metric function from plans to real values (plan costs) and W is a set of time windows.*

A solution for a planning model is called a *plan*. A plan is a sequence of grounded actions, $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ each with a respective time denoted by $Dispatch(a_i)$. The execution of a plan consists of a sequence of *happenings* corresponding to the effects of actions and exogenous effects in the world [13]. This sequence describes a trace of times, $t_{i=0\ldots k}$ and states, $s_{i=0\ldots k+1}$ such that $s_0 = I$ and for each $i = 0 \ldots k$, s_{i+1} is the result of executing the happening at time t_i from state s_i. The plan is valid if $s_{k+1} \models G$ (that is, the goal is satisfied in the final state reached by the plan).

We assume that the human's planning model Π^H, and planning agent's model Π^P share the same vocabulary, namely the same predicate symbols Ps, function symbols Vs, and actions As from the domain D, and objects Os from the problem. However, the action durations, conditions, and effects may be different, and the initial states I, goals G, and plan metric M may be different.

Even when a human and a planning agent have the same planning models $\Pi^H = \Pi^P$, there are typically multiple plans satisfying this planning model. Although a planner is intended to optimise the plan with respect to the plan metric, it is common to produce only one of the valid plans, rather than an optimal plan for a model. For some problems a planner might even fail to produce a plan at all. In part, this is an inevitable consequence of the undecidability of planning problems with numeric variables and functions [18], but it is also a consequence of the practical limits on the computational resources available to a planner (time and memory). These observations are equally valid for automated and human planners. In order to discuss the process of developing plan explanations, it is helpful to define the planning abilities of both the planner and the user. We model the planning capability of an agent as a partial function from planning models to plans:

Definition 2. *The* **planning capability** *of an agent A (human or machine), is a partial function, $\mathbb{C}^A$, from planning models to plans. Given the agent's planning model, Π^A, if $\mathbb{C}^A(\Pi^A)$ is defined, then it is a candidate plan π^A for the agent.*

The part of the function domain on which $\mathbb{C}^A$ is defined determines the planning competency of the agent – domain-problem pairs for which the agent cannot find a plan lie outside this competency. Note that the planning competency of an agent can be restricted by a bound on the computational resources the agent is allowed to devote to the problem, as well as by the capabilities of the agent in constructing and adequately searching the search space that the problem defines.

When A is an automated AI planner P, the computational ability is determined by the search strategy implemented in the planner and the resources allocated to the task. When A is a human planner H, the planning capability is determined by the understanding that the human has of the planning model and the patience and problem-solving effort they are willing to devote to solving the problem. It cannot be assumed that, if $\mathbb{C}^H(\Pi^H)$ is defined, that the human's model Π^H accurately reflects the world, or that the reasoning $\mathbb{C}^H$ is sound. This means that the plan may not be valid.

One aspect of the process of planning and explanation is that the user can revise their model Π^H as the process unfolds. However, it is also possible that the user can change their planning capability $\mathbb{C}^H$, by coming to a greater understanding of the model, by engaging in more reasoning, or by simply concluding that the solution provided by an automated system is satisfactory. It is also possible that the planner responses lead to the user changing their view of what might be a good plan to solve a problem, while still not adopting the solution offered by the planner.

2.2 Iterative Plan Exploration

We adopt the approach that the human user asks contrastive questions that impose additional restrictions ϕ on the agent's planning problem Π^P to generate a succession of hypothetical planning problems. The object of these questions and the resulting hypothetical plans is for the user to understand and ultimately arrive at a satisfactory plan. We call this process *Iterative Plan Exploration*.

Given the planning models Π^H and Π^P, and planning capabilities $\mathbb{C}^H$ and $\mathbb{C}^P$ of a human and planning agent, the two agents disagree when $\mathbb{C}^H(\Pi^H) \neq \mathbb{C}^P(\Pi^P)$, which can arise in the case that either of these terms is undefined, or if both terms are defined and yield different plans. We assume that the user is able to inspect the planner output and determine a question that will expose some part of the explanation for this difference. By questioning why certain decisions were made in the plan and receiving contrastive explanations the user can gain an initial understanding. As their understanding of the plan develops they can ask more educated questions to gain a deeper understanding or try to arrive at an alternative plan that they consider more satisfactory. This process concludes when the user is satisfied with some plan.

We formalise the iterative process of questioning and explanation as one of successive *model restriction*, in which the user asks contrastive questions in an attempt to understand the planning agent's plan and potentially steer the planning agent towards a satisfactory solution. We suppose that, when $\mathbb{C}^H(\Pi^H) \neq \mathbb{C}^P(\Pi^P)$, the user can construct a *foil*, ϕ, in the form of a constraint that $\mathbb{C}^P(\Pi^P)$ does not satisfy, so that seeking an explanation for the plan, $\mathbb{C}^P(\Pi^P)$, can be seen as seeking a plan for Π^P that also satisfies ϕ. This requirement acts as a restriction on Π^P and is captured as follows.

Definition 3. *A* **constraint property** *is a predicate, ϕ, over plans.*
A **constraint operator**, *$\times$ is defined so that, for a planning model Π and any constraint property ϕ, $\Pi \times \phi$ is a model Π', called a* **model restriction** *of Π, satisfying the condition that any plan for Π' is a plan for Π that also satisfies ϕ.*

The process in which the user interacts with a planner is an iterative one – the user successively views plans and seeks explanations by generating foils that impose additional restrictions on the planning problem. The collection of model restrictions forms a tree, rooted at the original model and extended by the incremental addition of new constraint properties, as shown in Figure 1. As the user inspects the result of applying $\mathbb{C}^P$ to a node in this tree, their own planning model and capability, Π^H and $\mathbb{C}^H$, may change, reflecting accumulating understanding of the plans that can be constructed for the model. It is worth emphasising that any constraint, ϕ, may be added to any model, so that the user is not forced to develop a tree of models in any particular way to arrive at the consequence of adding any specific constraint to a model.

It should be noted that, depending on the planning models and capabilities of the two participants, there might not exist any constraint achieving a common solution. For example, in the degenerate case in which $\mathbb{C}^P$ produces no plan at all, for any value of ϕ, then there can be no mutually satisfactory plan. Typically, the greater the differences between the planning models and capabilities of the two agents, the more likely it will be that there is no common satisfactory plan.

We formally capture the iterative process of model restriction and planning as:

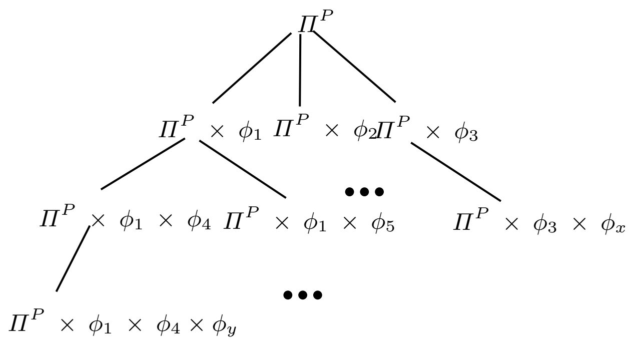
Π^P
$\Pi^P \times \phi_1$
$\Pi^P \times \phi_2$
$\Pi^P \times \phi_3$
$\Pi^P \times \phi_1 \times \phi_4$
$\Pi^P \times \phi_1 \times \phi_5$
$\Pi^P \times \phi_3 \times \phi_x$
$\Pi^P \times \phi_1 \times \phi_4 \times \phi_y$

Fig. 1. A fragment of a tree of model restrictions for a planner P. Each node n_i in the tree is a model restriction of the model of it's parent node n_{i-1}, and a constraint ϕ_i.

Definition 4. ***Iterative Model Restriction*** *For a planner P, and a user H: Let $\mathbb{C}^P$ and Π^P be the planner's underlying capability and planning model and $\mathbb{C}_0^H$ and Π_0^H be the initial capability and planning model of H. Let ϕ_i be the set of user imposed constraints, which is initially empty, i.e. $\phi_0 = \emptyset$. Each stage, i (initially zero), of this process starts with the planner producing a plan $\pi_i^P = \mathbb{C}^P(\Pi_i^P)$ for the model $\Pi_i^P = \Pi^P \times \phi_i$.*

The user responds to this plan π_i^P by potentially updating their capability and model to $\mathbb{C}_{i+1}^H$ and Pi_{i+1}^H and then either terminating the interaction, or asking a question that imposes a new constraint ϕ_{i+1} on the problem. This results in the planner solving a new constrained problem $\Pi_{i+1}^P = \Pi^P \times \phi_{i+1}$ at the next step.

We have assumed here that the planner's underlying capability and planning model $\mathbb{C}^P$ and Π^P do not evolve during the process. While this is not strictly necessary, possible evolution or improvement of the planner capabilities and model based on the sequence of user questions and the resulting ϕ_i is an issue we do not consider here. In contrast, the user's capability and planning model $\mathbb{C}^H$ and Π^H are assumed to evolve, but in unknown ways. Again, we do not attempt to model the user's learning process.

The exploration process could end in a variety of different ways. One way that the exploration can end is that the plan produced for the final model yields a plan that is acceptable to the user, so that the user adopts this plan for the original model. Alternatively the user, having explored the plans for several models, is persuaded in this process that the first plan produced by the planner for the original model is actually the desired plan. The exploration can also result in the selection of a plan somewhere between these two extremes, in which the user decides to adopt a plan produced for some intermediate model in the exploration. Finally, the exploration can end when the user explores the space and then rejects all of the plans the planner offers. In this case, the user might modify their planning model and capability as a consequence of what they observe and they might or might not conclude the process with a satisfactory plan for the original model. Krarup et al. [21] explore the hypothesis that the user will usually find value in the exploration and conclude in one of the three cases in which a mutually agreed plan is identified.

2.3 Temporal, Numeric, and Probabilistic Model Restrictions

The Iterative model restriction process presents a natural method to explore a space of plans through dialogue with the system, but does not succinctly represent more complex constraints in the space of solutions that arise from the interaction between numeric and probabilistic parts of the model. For instance, the trade-offs between two different rewards that the human user wishes to capture in the model's metric could be iteratively refined so that the user can see the outcome of different weightings between those rewards - but this is not an efficient way to represent what could be visually presented immediately with a Pareto-optimal set of solutions - if such a thing is possible to produce given the automated planner's capability.

In addition to this, our process does not account for uncertainties that might alter the plan during execution, which is a natural occurrence when executing a plan in the real world. While comparing different plans allows the user to understand the space of solutions to the planning instance, it might not reveal differences in how those plans might be realised in an uncertain environment.

3 Optimisation under Temporal Uncertainty

In many applications, the activities that need to be performed are already known, but there remains the problem of deciding when to perform those activities in order to meet constraints and optimise reward. As an example, consider the problem of 5G network slicing. Containerised components are hosted on pods in nodes within a data center (DC). Each pod is an allocation of a component's required share of resources. Multiple components can be linked to provide a service which satisfies the requirements defined in the service level agreement (SLA) reached between the provider and customer. However a number of events can occur which may result in the service configuration no longer being valid. Congestion at component input, for example, can result in packet drops such that the terms outlined in the SLA are no longer satisfied. Under such a scenario it may be necessary to reconfigure how the components are hosted within the DC. The decision of when to reroute traffic is influenced by two conflicting factors: the increased cost of migrating components early, and the risk associated with the probability distribution describing the SLA violation.

It is not known by the decision maker a-priori which combination of risk and cost is desired. The problem can be considered a bi-objective optimisation problem in which the solution is a Pareto optimal set of schedules optimising risk and cost. The relation to plan exploration is clear: the problem has many possible solutions with different characteristics, and the decision maker can benefit from exploring this space of solutions.

In this section we provide a background on the temporal network formalism and definitions of controllability. Then we discuss the Relaxable Chance Constrained Probabilistic Simple Temporal Network (r-cc-PSTN) which was introduced by Yu et al. [41]. We show that this can be expressed as a Joint Chance Constrained Optimisation Problem (JCCP) for which a rich suite of solution methods exist. Finally we discuss the potential for this to be incorporated within a bi-objective optimisation framework capable of generating the Pareto optimal set of schedules.

3.1 Temporal Networks and Controllability

Simple Temporal Networks (STN) [5] are used to represent temporal domains and reason about decisions under the influence of temporal constraints. An STN is a graph in which the nodes correspond to *time-points* and the edges (*links*) correspond to durations between the time-points. A solution to an STN is a schedule at which to execute a number of time-points such that the temporal constraints are satisfied. As such they are a natural formalism to represent scheduling problems such as 5G network slicing.

Simple Temporal Networks with Uncertainty (STNU) were introduced [39] to capture uncertainty in the problem through the inclusion of set-bounded *contingent links*, over which the operator has no control. In STNU semantics, a distinction is made between *contingent links*, for which the duration of the interval is uncertain and *requirement links* for which we can choose the duration.

Definition 5 (STNU). *A STNU is a tuple, $S^U = \langle T_c, T_u, C, G \rangle$ where $b_1, b_2, ..., b_B \in T_c$ is the set of controllable time-points and $e_1, e_2, ..., e_E \in T_u$ is the set of uncontrollable time-points, such that $t_1, t_2, ..., t_n \in \{T_c \cup T_u\}$. The set C, is the set of temporal requirement constraints between two time-points, normally written in the form $c(t_j, t_i) = t_j - t_i \in [l^c_{ij}, u^c_{ij}]$. The set G is the set of contingent links given in the form $g(e_i, b_i) = e_i - b_i \in [l^g_i, u^g_i]$. Here, $l^{c \vee g}, u^{c \vee g}$ denote the lower and upper limits for the constraint or contingent link respectively. Let $s\,(b) \in \mathcal{R}^+$ be the assignment of a value to the controllable time-point b. Let $o\,(e) \in \mathcal{R}^+$ be the value observed by an uncontrollable time-point e. A projection of a contingent link g_i is $\omega_i := o(e_i) - s(b_i)$.*

The challenge in scheduling STNUs lies in the fact that the set of contingent links may take any random value within their bounds, and therefore an effective execution strategy must consider all possible projections for each contingent link.

When dealing with uncertainty in temporal networks it is typical to classify the problem in terms of controllability [40], which can be considered as a way of classifying how much control the agent has over the outcome of the network [40]. Controllability of an STNU is typically separated into 3 categories (strong, dynamic, weak). In a strongly controllable network, there exists an assignment to all controllable time-points that can be determined a-priori and will satisfy all constraints no matter the outcome of the contingent links.

Definition 6 (Strong Controllability). *Denote Ω, the space of projections of the contingent links: $\Omega = \times_{g \in G}[l^g, u^g]$. Let the schedule δ, be the assignment $s\,(b)$, $\forall b \in T_c$. An STNU S is said to be strongly controllable if: $\exists \delta \mid \forall \omega \in \Omega$, δ satisfies all constraints.*

We denote: $C_u \subseteq C$, the set of *uncontrollable constraints* containing an uncontrollable time-point. One can substitute $e = b + \omega$ for the contingent link preceding/succeeding the uncontrollable constraint. To check whether an STNU, S_U is SC, it is sufficient to check that the uncontrollable constraints are satisfied for the worst possible projection of the contingent links. i.e. $\min\{c \mid \omega \in [l^g, u^g]\} \geq l^c$, $\max\{c \mid \omega \in [l^g, u^g]\} \leq u^c$ for every $c \in C_u$.

Where sufficient data is available, it is often more representative to model the space of possible projections of a contingent link by a probability density function [36, 11], creating a Probabilistic Simple Temporal Network (PSTN). This allows the scheduling process to focus on the durations *most likely* to be realised at execution.

Definition 7 (PSTN). *A PSTN is a tuple, $S^P = \langle T_c, T_u, C, D\rangle$, where T_c, T_u and C are as per the STNU. The set of probabilistic constraints, D are in the form $d(e_i, b_i) = e_i - b_i = X_i$, where X_i is a random variable with a set of outcomes Ω_i, probability density function $f(\omega_i)$ and cumulative probability function $F(\omega_i)$.*

It should be noted that it is often impossible to find a SC schedule robust to all possible outcomes of an unbounded distribution. As a result, it is typical to squeeze the distribution by neglecting the extreme, unlikely outcomes in the tails of the distributions, i.e: $\Omega_i^* = [l_i^d, u_i^d]$. We denote $d^*(e_i, b_i)$, the restricted probabilistic constraint, and performing this restriction transforms the probabilistic constraint to a contingent link, i.e. $d^*(e_i, b_i) = e_i - b_i = X_i^* \in [l_i^d, u_i^d] \equiv g(e_i, b_i)$. Applying this transformation to all $d \in D$ is equivalent to transforming the PSTN, S^P to an equivalent STNU, S^{U*}. However the schedule is now only robust to the outcomes considered in S^{U*}. The probability mass excluded by performing this transformation is the *risk* of S^P.

Definition 8 (Robustness and Risk). *We denote $\Omega_R \subseteq \Omega$ and $c(\omega)$ the value of each constraint $c \in C$ given an outcome ω. If $\omega \in \Omega$ and for every $c \in C$, $c(\omega) \in [l_{ij}^c, u_{ij}^c]$: then $\omega \in \Omega_R$. The robustness R, is $P(\Omega_R)$, while the risk Δ, is $P(\bar{\Omega}_R)$, where $\bar{\Omega}_R$ denotes the complement of the set Ω_R.*

Since the joint probability function $P(\bar{\Omega}_R)$ is non-trivial, it is typical to treat uncontrollable constraints independently and bound above the risk using Boole's inequality. The risk can of then be approximated through: $\Delta = \sum_i^{|C_u|} F(l_i^d) + (1 - F(l_i^d))$. The values of u_i^d and l_i^d are determined through the SC relationships outlined previously, through substituting l^g, u^g for l^d, u^d.

3.2 PSTN Strong Controllability and Risk in Literature

Tsamardinos [36] takes a risk minimisation approach to PSTN SC, and makes use of various assumptions to leverage Sequential Quadratic Programming. Likewise, Santana et al. [32] and Lund et al. [24] make varying assumptions to permit the use of LPs to allocate risk in PSTNs. Fang et al. [11] introduced the notion of chance constrained PSTNs (cc-PSTN); by enforcing an allowable tolerance on the risk as a constraint in the system, such as $\Delta \leq \alpha$, where $\alpha \in [0, 1]$. Some other objective function could then be optimised, while ensuring that the schedule risk does not exceed α.

In some instances the risk required to enforce SC can be deemed too high. Yu et al. [41] extended the chance-constrained framework to the *relaxable* chance constrained probabilistic simple temporal network (r-cc-PSTN) by permitting the use of soft constraints which can be relaxed. A cost is then paid relative to the amount of relaxation. This enables solutions to be found for over-constrained cc-PSTNs. The r-cc-PSTN is very general, and is solved by Yu et al. using a nonlinear solver, combined with a conflict detection mechanism based on identification of negative cycles in STNUs. Nonlinear optimization problems are solved to eliminate these cycles.

To the best of the authors' knowledge, all previous SC approaches for PSTNs either use Boole's inequality to bound above the risk [11, 41, 32, 24], or solve a generic non-linear optimisation problem [36]. Using Boole's inequality permits the use of Linear Programming solvers, however it can be overly conservative - particularly when the number of uncontrollable constraints is large. Whereas posing the problem in a generic non-linear setting can either be computationally expensive or offer no guarantee of global optimality.

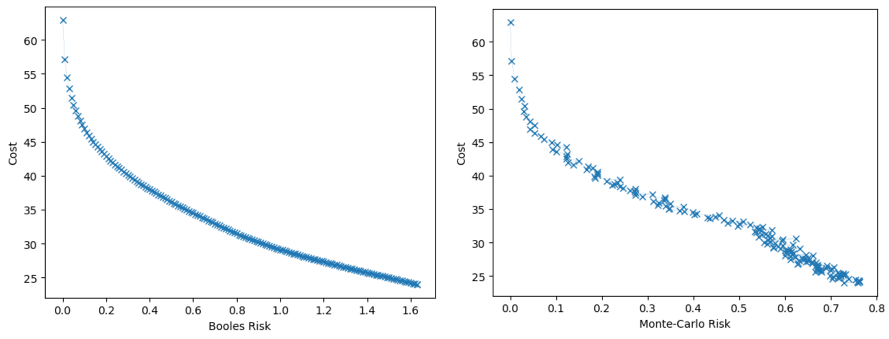

Fig. 2. Figure showing the Pareto optimal front generated using Boole's inequality (left) in comparison to the equivalent Monte-Carlo risk (right)

In Figure 2, an example r-cc-PSTN was solved for strong controllability with Boole's inequality bounding the risk. Following this, the schedules obtained for each cost were simulated using a Monte-Carlo execution approach enabling the cost to be plotted against the true risk. As can be seen, the Pareto front obtained using Boole's inequality is not guaranteed to be the true Pareto optimal solution for cost and risk. Likewise as the number of uncontrollable constraints increases, the Boole's risk is not guaranteed to be bounded within $[0, 1]$. As such it is not interpretable and gives little useful information to the human required to reason over the Pareto front.

3.3 On Pareto Optimal Schedules to PSTNs

In this section we discuss in greater detail the r-cc-PSTN [41] and highlight how these can be solved as a Joint Chance Constrained Optimisation Problem (JCCP) enabling the evaluation of true Pareto optimal solutions.

Definition 9 (r-cc-PSTN). *A r-cc-PSTN is a tuple $S^R = \langle T_c, T_u, C_c, C_u, D, W, \alpha \rangle$, where T_c, T_u and D are as per the definition of PSTN. The set of requirement constraints C is partitioned into a set of controllable constraints C_c: $c(b_j, b_i) = b_j - b_i \in \{l_{ij}^c, u_{ij}^c\}$ and uncontrollable constraints C_u: $c(e_j, b_i) = e_j - b_i \in \{l_{ij}^c\ u_{ij}^c\} \vee c(b_j, e_i) = b_j - e_i \in \{l_{ij}^c, u_{ij}^c\}$. There exists some subset $C_{c,s} \subseteq C_c$ and $C_{u,s} \subseteq C_u$ which are considered soft constraints. We introduce the lower and upper relaxation variables: $\check{r}_{ij}, \hat{r}_{ij} \in \mathcal{R}^+$ for each constraint in $\{C_{c,s} \cup C_{u,s}\}$: $c(t_j, t_i) = t_j - t_i \in \{l_{ij}^c - \check{r}_{ij}, u_{ij}^c + \hat{r}_{ij}\}$. The* *relaxation weight $w \in W$, associated with relaxing constraint $c \in \{C_{c,s} \cup C_{u,s}\}$ is the relative cost of relaxing the constraint by one time unit, such that the relaxation cost k is calculated as the linear sum: $k = w\check{r} + w\hat{r}$. Finally, $\alpha \in [0, 1]$ is the risk bound representing the maximum allowable probability of failure across all $c \in C_u$.*

To elucidate the key characteristics of such a problem, we return to the motivating example of 5G slicing. First we describe an example situation (Figure 3), and then the example PSTN problem that results (Figure 4). In Figure 3, we consider two nodes. A

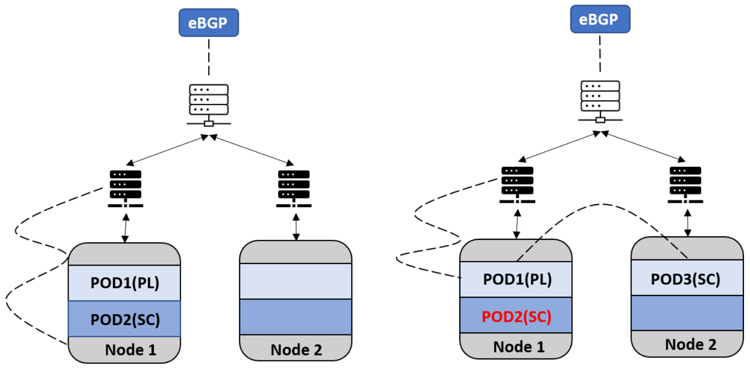

Fig. 3. Diagram showing data center configuration before (left) and after (right) migrating traffic.

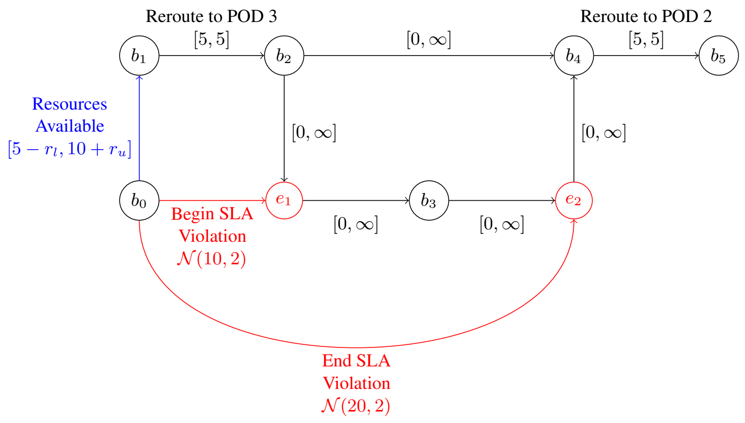

Fig. 4. Example PSTN showing data center problem.

service consisting of two components, payload (PL) and service controller (SCtrl), is hosted in the DC. Traffic is processed by PL and then by SCtrl before exiting through the exit gateway. We consider three conditions outlined in the SLA: latency of the path, the percentage of packet drops, and the cost of hosting the pods on the nodes. The rental of Node 2 is higher than that of Node 1, and the latency of the path POD1 $\rightarrow$ POD3 is larger than that of POD1 $\rightarrow$ POD2. Initially, PL and SCtrl are hosted in POD1 and POD2 of Node1 such that the terms in the SLA are met.

At some point in the day, increased traffic to the service requires a scaled up SCtrl for which there is insufficient resources available in the current configuration using POD2. If SCtrl continues to be hosted in Node 1 POD2, then packets cannot be processed at the rate of arrival, resulting in increased packet drops and consequently a violation of the SLA. The penalty associated with packet drops is greater than the increased contributions from latency and the cost of rental. The decision is made to spin off another pod (POD3) in Node 2 before the SLA violation occurs and reroute the traffic from SCtrl in Node 1 to PL in Node 2. The traffic should then be rerouted back to the original configuration after the violation has ended. We assume that moving the component from "Node 1 POD2" to "Node 2 POD3" and back both take 5 time units. The SLA violation due to the congestion can take place any time in the future but with a distribution of $\mathcal{N}(10, 2)$. The end of the SLA violation can also be described with a probability distribution of $\mathcal{N}(20, 2)$.

The time at which to move the component is constrained by the availability of resources in Node 2, modelled by the constraint $b_0 \rightarrow b_1$ in Figure 4. Between 5 and 10 units after b_0, there exists sufficient resources for PL to be spun on Node 2. The ideal decision would be to move the component as early as possible to minimise the probability that the SLA violation penalty will be incurred. We can choose to schedule $b_1 = 5$, however this means that the component will not be activated on Node 2 until $b_2 = 10$ and thus the resulting strongly controllable schedule will have only a 50% chance of success (if the SLA begins prior to the mean of 10 units, the penalty will be incurred).

The availability of resources is a soft constraint that we can relax from $[5, 10]$ to $[0, 10]$ incurring some cost associated with relocating existing components. This relaxation allows b_1 to be scheduled earlier, decreasing the risk of the SLA violation. The relaxation cost is a function of the amount by which we relax the constraint. The optimal scheduling decision therefore becomes a trade-off between the relaxation cost and the risk associated with incurring the SLA violation cost.

To solve this problem, a bi-objective optimisation framework such as the ϵ-constraint method [4] can be used to generate the Pareto optimal front minimising both risk and cost. Within this framework, one of the objectives (i.e. the risk) can be treated as a constraint by imposing a limit which is subsequently varied and the other objective (i.e. cost) is optimised until the problem becomes infeasible. The underlying problem of minimising cost subject to the constraint on risk can be considered as a JCCP: $\min_x \{c^T x \mid Ax \leq b, P(Tx + q \geq \xi) \geq 1 - \alpha\}$.

With some algebraic manipulation, r-cc-PSTNs can quite easily be expressed in this form. The decision vector $x \in \mathcal{R}^n$, would be the controllable time-points comprising the schedule, combined with the relaxation variables for each soft constraint. The controllable constraints can be encapsulated in the linear inequality: $Ax \leq b$, where $A \in \mathcal{R}^{m \times n}$ are the constraint coefficients and $b \in \mathcal{R}^m$ are the upper bounds. Similarly, the uncontrollable constraints can be captured in the joint chance constraint: $P(Tx + q \geq \xi) \geq 1 - \alpha$, where $T \in \mathcal{R}^{p \times n}$ is the matrix of coefficients and $q \in \mathcal{R}^p$ are the bounds. Here, ξ is a p dimensional random variable with mean vector, μ and covariance matrix, Σ and $\alpha \in [0,1]$ is the joint bound on the probability of failure. Finally any linear objective can be implemented within the objective function $c^T x$, however we consider that we wish to minimise the relaxation cost and thus $c \in \mathcal{R}^n$, is the vector of relaxation weights w associated with each relaxation variable.

Prekopa [27, 28] proved that if the probability distribution is log-concave, then the cumulative probability function $F(z) = P(\xi \leq z)$ is also log-concave and thus the set $\{z \mid -\log(F(z)) \leq -\log(1-\alpha)\}$ is convex. Many interesting distributions contain this characteristic [29]. The result is that r-cc-PSTN SC as JCCP is a convex optimisation problem with a tractable evaluation of the global optimal schedule. More detail can be found in a recent survey of solution methods [37, 10] and overview on the topic [30].

4 Discussion and Conclusion

In this section we briefly discuss three possible directions for future work that could combine the approaches from Sections 2 and 3. A naive approach would be to use iterative plan exploration to generate a plan $\pi = \langle a_1, a_2, \ldots, a_n \rangle$, whose *happenings* become the nodes of a PSTN. That PSTN could then be analysed for SC. The limitation of this naive approach is that the plan exploration would not benefit from the additional insight provided by the PSTN SC analysis. Below we describe three alternatives for closer integration of the approaches.

Constrastive Explanations of Strongly Controllable Plans. In iterative plan exploration, model restrictions are used to generate a hypothetical plan that embodies the "what if" question posed by the user. Plans are compared against one another as a form of explanation – the aim of which is to make explicit the impact of their suggestions. In the user study carried out by Krarup et al. [21], plan metrics were directly compared, assuming a single realisation of the plan's time-points that was selected by the planner. Instead, the approach presented in Section 3 could be used to provide a more in-depth comparison between the two plans by comparing Pareto-optimal schedules for their respective actions.

Iterative Restrictions to PSTNs. A direction for future work in explainable scheduling would be to apply the paradigm of iterative plan exploration to the problem of PSTN SC. Just as it is the case that the decision maker might have preferences that are not completely captured by the planner's model, it can also be the case that the Pareto optimal frontier does not actually represent the complete set of solutions of interest. By allowing the user to apply restrictions as new constraints in the PSTN the user will be able to see the impact of restrictions that iteratively force solutions away from the Pareto-optimal frontier. Just as in iterative plan exploration, this process could converge in a schedule that better adheres to the user's true preferences, or increase their trust in the original set of solutions.

Plan Exploration in Discrete and Continuous Spaces. Krarup et al. [21] showed that by using the most common restrictions requested by users – action inclusion, exclusion, ordering, and temporal constraints – iterative model restriction can combine these constraints to encapsulate more specific questions and eventually converge to any valid plan. However, using successive queries to explore a continuous space such as real-valued cost or risk would be very inefficient. This limitation could be tackled by embedding the analysis provided by the PSTN SC analysis into the plan exploration tools developed by Krarup et al. This would extend the common restrictions above to also include constraints on cost and risk that are drawn from observation of the Pareto-optimal set of solutions. Unlike the *iterative restrictions to PSTNs* described above, these constraints would be applied to a planning model in the current exploration tree and used to generate a new plan that could potentially have a different set of actions. The result would be that the user could efficiently explore the different trade-offs exhibited by a variety of possible plans, before converging upon a chosen plan and schedule.

References

1. Bercher, P., Biundo, S., Geier, T., Hoernle, T., Nothdurft, F., Richter, F., Schattenberg, B.: Plan, Repair, Execute, Explain-How Planning Helps to Assemble your Home Theater. In: ICAPS (2014)
2. Chakraborti, T., Sreedharan, S., Zhang, Y., Kambhampati, S.: Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In: IJCAI (2017)
3. Chakraborti, T., Sreedharan, S., Kambhampati, S.: The Emerging Landscape of Explainable AI Planning and Decision Making. In: IJCAI. pp. 4803–4811 (2020)
4. Chircop, K., Zammit-Mangion, D.: On-constraint based methods for the generation of pareto frontiers. Journal of Mechanics Engineering and Automation **3**(5), 279–289 (2013)
5. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artificial intelligence **49**(1-3), 61–95 (1991)
6. Edelkamp, S.: On the compilation of plan constraints and preferences. In: ICAPS. pp. 374–377 (2006)
7. Eifler, R., Cashmore, M., Hoffmann, J., Magazzeni, D., Steinmetz, M.: A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning. In: AAAI. pp. 9818–9826 (2020)
8. Eriksson, S., Helmert, M.: Certified Unsolvability for SAT Planning with Property Directed Reachability. In: ICAPS. vol. 30, pp. 90–100 (2020)
9. Eriksson, S., Röger, G., Helmert, M.: Unsolvability certificates for classical planning. In: AAAI (2017)
10. Fábián, C.I., Csizmás, E., Drenyovszki, R., van Ackooij, W., Vajnai, T., Kovács, L., Szántai, T.: Probability maximization by inner approximation. Acta Polytechnica Hungarica **15**(1), 105–125 (2018)
11. Fang, C., Yu, P., Williams, B.C.: Chance-constrained probabilistic simple temporal problems. In: AAAI (2014)
12. Fox, M., Long, D.: Extending the exploitation of symmetries in planning. In: ICAPS. pp. 83–91 (2002)
13. Fox, M., Long, D.: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. JAIR **20**, 61–124 (2003)
14. Fox, M., Long, D., Magazzeni, D.: Explainable planning. In: IJCAI-17 workshop on Explainable AI (2017)
15. Gerevini, A., Long, D.: Plan constraints and preferences in pddl3. Tech. rep., Department of Electronics for Automation (2005)
16. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: theory and practice. Elsevier (2004)
17. Göbeldecker, M., Keller, T., Eyerich, P., Brenner, M., Nebel, B.: Coming up with good excuses: What to do when no plan can be found. In: Dagstuhl Seminar Proceedings (2010)
18. Helmert, M.: Decidability and undecidability results for planning with numerical state variables. In: AIPS. p. 44–53 (2002)

19. Kasenberg, D., Roque, A., Thielstrom, R., Chita-Tegmark, M., Scheutz, M.: Generating justifications for norm-related agent decisions. In: arXiv preprint arXiv:1911.00226 (2019)
20. Kim, J., Muise, C., Shah, A., Agarwal, S., Shah, J.: Bayesian inference of linear temporal logic specifications for contrastive explanations. In: IJCAI. pp. 5591–5598 (2019)
21. Krarup, B., Krivic, S., Lindner, F., Long, D.: Towards Contrastive Explanations for Comparing the Ethics of Plans. In: ICRA-20 Workshop on Against Robot Dystopias (2020)
22. Krarup, B., Krivic, S., Magazzeni, D., Long, D., Cashmore, M., Smith, D.E.: Contrastive explanations of plans through model restrictions. JAIR pp. 533–612 (2021)
23. Kulkarni, A., Zha, Y., Chakraborti, T., Vadlamudi, S.G., Zhang, Y., Kambhampati, S.: Explicablility as minimizing distance from expected behavior. In: arXiv preprint arXiv:1611.05497 (2016)
24. Lund, K., Dietrich, S., Chow, S., Boerkoel, J.: Robust execution of probabilistic temporal plans. In: AAAI (2017)
25. McDermott, D.: The 1998 AI planning systems competition. AI Magazine **21**(2), 35–55 (1998)
26. Mueller, S.T., Hoffman, R.R., Clancey, W.J., Emrey, A., Klein, G.: Explanation in Human-AI Systems: A Literature Meta-Review, Synopsis of Key Ideas and Publications, and Bibliography for Explainable AI. CoRR **abs/1902.01876** (2019)
27. Prékopa, A.: Logarithmic concave measures with applications to stochastic programming. Acta Scientiarum Mathematicarum **32** (1971)
28. Prékopa, A.: On logarithmic concave measures and functions. Acta Scientiarum Mathematicarum **34** (1973)
29. Prékopa, A.: Probabilistic programming. Handbooks in operations research and management science **10**, 267–351 (2003)
30. Prékopa, A.: Stochastic programming, vol. 324. Springer Science & Business Media (2013)
31. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
32. Santana, P., Vaquero, T., Toledo, C., Wang, A., Fang, C., Williams, B.: Paris: A polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty. In: ICAPS (2016)
33. Smith, D.: Planning as an iterative process. In: AAAI (2012)
34. Sreedharan, S., Chakraborti, T., Kambhampati, S.: Handling model uncertainty and multiplicity in explanations via model reconciliation. In: ICAPS (2018)
35. Sreedharan, S., Srivastava, S., Smith, D., Kambhampati, S.: Why couldn't you do that? explaining unsolvability of classical planning problems in the presence of plan advice. In: arXiv preprint arXiv:1903.08218 (2019)
36. Tsamardinos, I.: A probabilistic approach to robust execution of temporal plans with uncertainty. In: Hellenic Conference on Artificial Intelligence. pp. 97–108. Springer (2002)
37. Van Ackooij, W.: A discussion of probability functions and constraints from a variational perspective. Set-Valued and Variational Analysis **28**(4), 585–609 (2020)
38. Van Melle, W.: Mycin: a knowledge-based consultation program for infectious disease diagnosis. International journal of man-machine studies **10**(3), 313–322 (1978)
39. Vidal, T., Ghallab, M.: Dealing with uncertain durations in temporal constraint networks dedicated to planning. In: ECAI (1996)
40. Vidal, T., Fargier, H.: Handling contingency in temporal constraint networks: From consistency to controllabilities. JAIR **11**, 23–45 (01 1999). https://doi.org/10.1080/095281399146607
41. Yu, P., Fang, C., Williams, B.: Resolving over-constrained probabilistic temporal problems through chance constraint relaxation. In: Proceedings of the AAAI Conference on Artificial Intelligence (2015)