Explaining Goal Conflicts in Oversubscription Planning

A dissertation submitted towards the degree
Doctor of Natural Sciences
of the Faculty of Mathematics and Computer Science
of Saarland University

by

Rebecca Eifler

Saarbrücken, 2025

SAARLAND UNIVERSITY

Day of Colloquium	24.07.2025
Dean of Faculty	Prof. Dr. Roland Speicher
Chair of the Committee	Prof. Dr. Jens Dittrich
Reviewers	Prof. Dr. Jörg Hoffmann
	Prof. Dr. Holger Hermanns
	Prof. Sylvie Thiébaux PhD
Academic Assistant	Dr. Daniel Höller

ABSTRACT

Many real-world planning scenarios are characterized by oversubscription problems such as logistics with limited vehicle capacity and fuel restrictions, rover mission planning with time and energy constraints, but also your weekly activity and housework plans with insufficient free time. As a result, not all goals within the given task can be satisfied. Conventional approaches assume global optimization objectives, but it is often difficult to identify such objectives, and they are frequently in conflict with one another. An iterative planning approach is more suitable, wherein users consider sample plans and refine their preferences based on these plans. In such a setting, it is crucial to provide not only the plans themselves, but also explanations elucidating the conflicts between goals, preferences and objectives. This facilitates the user's understanding and enable them to identify satisfactory trade-offs.

To this end, we introduce a form of contrastive explanation. We support user questions of the form "Why is Q not achieved by the plan?", by providing explanations through goals A that conflict with Q, i.e. "To satisfy Q you have to forgo A". We develop a set of algorithms that enable this form of explanation by computing the minimal unsolvable goal subsets based on goal space search and state space search. The evaluation of these algorithms shows that the required analysis in terms of scalability is comparable to that of oversubscription planning. Additionally, we conducted a large user study with crowdworkers (N=100 in each of 3 domains) to evaluate our framework. We compared users with and without access to explanations and found that the explanations enabled users to better identify trade-offs, indicating a better understanding of the planning task.

Conflicts often arise due to resource or time constraints. We address the follow-up question "Why is A in conflict with Q?" by providing explanations based on the minimum relaxations of constraints under which the conflict resolves. We investigate two approaches to computing such explanations: specialized algorithms and compilation. A basic algorithm involves simply looping over all relaxations and computing the conflicts for each relaxation separately. We improve over this with two algorithms that exploit information such as reachable goal subsets and states across relaxations. Alternatively, we explore a compilation using specialized soft goals that identify each relaxation.

In order to provide valuable explanations, it is essential that explanations cover aspects of the plans in which the user is interested. However, users often find it difficult to formalize their preferences. Therefore, we explore the potential of learning preferences from example plans, focusing on a single preference at a time and requesting that the user rates examples as either good or bad. Based on prior work on learning LTL_f formulas, we then extract a preference from these examples. We conduct an empirical study of this approach in a classical planning setting, using hidden target formulas to simulate user preferences.

ZUSAMMENFASSUNG

Viele reale Planungsszenarien sind durch Überschreibungsprobleme gekennzeichnet, z. B. die Logistik mit begrenzter Fahrzeugkapazität und Treibstoffbeschränkungen, die Planung von Rover-Missionen mit Zeit- und Energiebeschränkungen, aber auch die Planung der wöchentlichen Aktivitäten und Hausarbeiten bei unzureichender Freizeit. Infolgedessen können nicht alle Ziele erfüllt werden. Herkömmliche Ansätze gehen von globalen Optimierungszielen aus, doch ist es oft schwierig, solche Ziele zu identifizieren, und sie stehen häufig in Konflikt zueinander. Besser geeignet ist ein iterativer Planungsansatz, bei dem die Benutzer Musterpläne betrachten und ihre Präferenzen auf der Grundlage dieser Pläne verfeinern. Bei einem solchen Ansatz ist es von entscheidender Bedeutung, nicht nur die Pläne selbst, sondern auch Erklärungen zu den Konflikten zwischen Zielen, Präferenzen bereitzustellen. Dies erleichtert das Verständnis des Nutzers und versetzt ihn in die Lage, zufriedenstellende Kompromisse zu finden.

Zu diesem Zweck führen wir eine Form der kontrastiven Erklärung ein. Wir unterstützen Nutzerfragen der Form “Warum wird Q nicht durch den Plan erreicht”. Wir beantworten diese in Form von Zielen A, die mit Q in Konflikt stehen, d.h.“Um Q zu erfüllen, muss man auf A verzichten”. Wir entwickeln eine Reihe von Algorithmen, die diese Form der Erklärung ermöglichen, indem sie die minimalen unlösbaren Zielteilmengen auf der Grundlage von Zielraumsuche und Zustandsraumsuche berechnen. Die Evaluierung dieser Algorithmen zeigt, dass die erforderliche Analyse in Bezug auf Skalierbarkeit sich ähnlich wie Überschreibungsprobleme verhält. Darüber hinaus führen wir eine große Crowd-Worker-Benutzerstudie durch (N=100 in jeder von 3 Domänen), in der wir unser Framework evaluieren. Im Vergleich zwischen Nutzern mit und ohne Zugang zu den Erklärungen stellen wir fest, dass die Erklärungen es den Nutzern ermöglicht, Kompromisse besser zu erkennen, was auf ein besseres Verständnis der Planungsaufgabe hindeutet.

Konflikte entstehen oft aufgrund von Ressourcen- oder Zeitmangel. Wir adressieren die Folgefragen “Warum ist A im Konflikt mit Q?” mit Erklärungen, die auf den minimalen Relaxierung der Beschränkungen basieren, unter denen die Konflikte verschwinden. Wir untersuchen zwei Ansätze zur Berechnung solcher Erklärungen: spezialisierte Algorithmen und Kompilierung. Der Basis-Algorithmus besteht darin, über alle Relaxierungen zu iterieren und die Konflikte für jede einzeln zu berechnen. Wir verbessern diesen Ansatz mit zwei Algorithmen, die Informationen wie erreichbare Zielteilmengen und Zustände über die Relaxierungen hinweg nutzen. Alternativ dazu erforschen wir eine Kompilierung unter Verwendung spezialisierter weicher Ziele, die die Relaxierungen identifizieren.

Um nützliche Erklärungen geben zu können, ist es wichtig, dass die Erklärungen die Aspekte der Pläne abdecken, an denen der Nutzer interessiert ist. Allerdings fällt es den Nutzern oft schwer, ihre Präferenzen zu formalisieren. Daher untersuchen wir das Potenzial des Lernens von Präferenzen anhand von Beispielplänen. Wir konzentrieren uns jeweils auf eine einzelne Präferenz und fordern den Nutzer auf, Beispiele entweder als gut oder schlecht zu bewerten. Basierend auf früheren Arbeiten zum Lernen von LTL_f-Formeln extrahieren wir dann eine Präferenz aus diesen Beispielen. Wir führen eine empirische Studie dieses Ansatzes in einer klassischen Planungsumgebung durch, wobei wir versteckte Zielformeln verwenden, um Benutzerpräferenzen zu simulieren.

ACKNOWLEDGMENTS

First, I would like to express my gratitude to Jörg Hoffmann for offering me the opportunity of pursuing my PhD studies. I appreciate his encouragement and support, and his patience with my struggles. I am grateful that he introduced me to Explainable Planning and the ICAPS community.

I would like to thank Holger Hermanns for agreeing to review this thesis. Spacial thanks go to Sylvie Thiébaux, for reviewing this thesis but also for her patience and all the opportunities and support she gave me.

I would also like to thank my colleagues from the FAI group for the great time, the discussions and the competitions we had: Álvaro, Chaahat, Dan, Daniel, Daniel, Jan, Julia, Marcel, Marcel, Max, Pascal, Thorsten and Timo. Thanks to Julia for our office conversations and Dan for the great conversations during our walks and that his office door was always open for me. Thanks to Dan and Merlin for proofreading this thesis.

I would like to take this opportunity to thank Jeremy Frank for his support, great ideas and insightful comments, and for the opportunity to present our work at NASA.

I would like to thank my family. My parents Petra and Tankred and my sister Marie, who have always supported me and believed in me.

Last but not least, I would like to thank Merlin. Without him, this thesis would not have been possible. He was a software consultant, a discussion partner, a calming influence when I had doubts, the best support you can have.

Thank you!

Table of Contents

CHAPTER 1

INTRODUCTION

Wieso, weshalb, warum?
Wer nicht fragt bleibt dumm.
- Sesamstraße

Why,why,why?
Those who don't ask remain stupid.
- Sesame Street

People are increasingly relying on *artificial intelligence* (AI) to solve various tasks. However, effective and satisfying collaboration between a human and an AI system requires much more than mere performance. It is crucial to understand and trust the AI's decisions. In recent years, *explainable* AI (XAI) [Speith, 2022, Chakraborti et al., 2020, Gunning and Aha, 2019, Horizon Europe, 2021] has gained prominence.

A promising candidate for safe, robust, and trustworthy AI is *automated planning*, a subfield of AI that aims to find a plan to solve a task based on a *model*. Such a model defines the possible states of the task and the actions that can be used to transition from one state to another. Model-based approaches are well-suited for explanations due to their explicit representation of world knowledge and the inference methods, which facilitate access to causal information.

Planning is typically a non-collaborative process. Usually, the user defines the model including the "goal" and the planning system, if possible, provides a plan. However, according to Smith [2012], this approach is often not applicable in real-world scenarios. Not all goals are always known in advance, and the user may also have preferences, e. g. based on experience, as to how these goals should be achieved. In addition, it is often not possible to satisfy all goals and preferences. This necessitates finding a compromise, for which an *iterative process* is more suitable. This process enables the user to refine their preferences based on sample plans and facilitates finding and agreeing on a satisfactory compromise.

In this context, it is essential to provide explanations for users' questions concerning the sample plans, goals, and preferences. These explanations should help users in better understanding the dependencies between the goals and their preferences. Users might ask questions such as: "Why does the sample plan not satisfy goal `x`?" Such *contrastive*

questions are a common way to ask for clarification. The user expects a goal to be satisfied or wants it to be satisfied and therefore asks, “Why not `x`?”. A potential answer should explain the implication of the alternative `x` specified by the user.

We propose a framework for providing contrastive explanations based on the concept of minimal goal conflicts. The goals represent aspects of plans that the user is interested in. These include goals such as “take an image of the crater”, but also more complex properties like “the rover should not take two images in a row, as this can overheat the camera module”, which reflect the user’s preferences. Conflicts can be used to provide answers such as “If you want to take an image of the crater, you have to take two images in a row”. This explains to the user what they must forgo in order to satisfy their alternative.

The work is divided into three parts each of which addresses a distinct objective of the framework. In Part I, we discuss how to identify and use goal conflicts to provide contrastive explanations. How to explain why a goal conflicts exists and how to resolve them, is addressed in Part II. Finally, Part III examines how to find suitable properties to provide explanations for aspects in which the user is interested.

1.1 Illustrative Example

Throughout this work, we utilize the planning of a Mars rover mission by a group of researchers as a running example. This Mars rover mission is modeled by a simple planning task. Figure 1 illustrates the map of a simple instance. There are four targets: a single crater, a rock, a group of three craters and an area covered with ice.

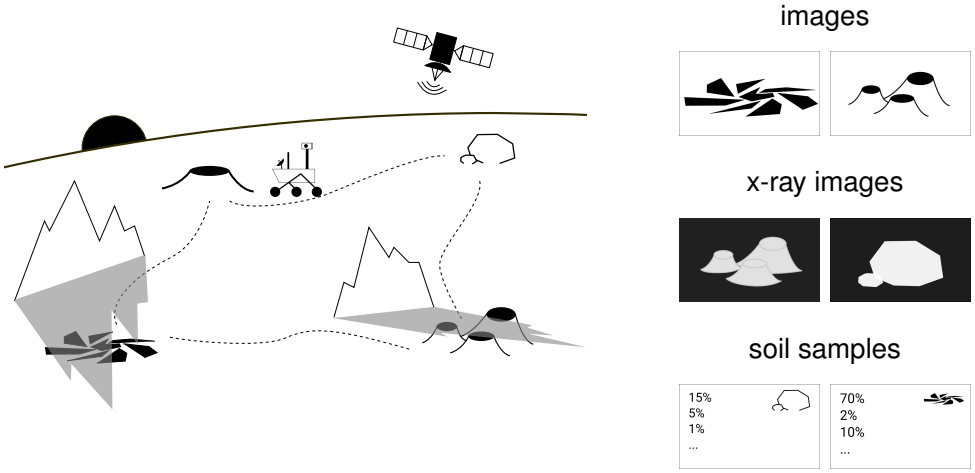


Figure 1: Map and data samples of the Mars Rover Mission.

The data samples scheduled for this mission consist of normal images of the crater group and the ice, X-ray images of the rock and the crater group and soil samples of the ice and the rock. After collecting the data, the rover must upload the data to a relay satellite. The rover can perform the following actions:

- drive between connected locations `l1` and `l2`: `drive(l1,l2)`
- collect a soil sample at location `l`: `soil-sample(l)`

- take an image at location `l`: `image(l)`
- take an x-ray image at location `l`: `x-ray-image(l)`
- upload data point `d` to relay satellite : `upload(d)`
- idle (do nothing): `idle`

A solution for a particular instance is defined as an action sequence specifying the order in which locations are traversed by the rover, the data samples to be collected at each location and when they are to be uploaded.

At first glance, collecting and uploading the requested data points seems straightforward. However, the following constraints must be taken into account, as illustrated in Figure 2. The rover only has limited energy and data storage. Furthermore, to take images, the corresponding location must be sufficiently illuminated, and for data upload, the relay satellite must be within signal range.

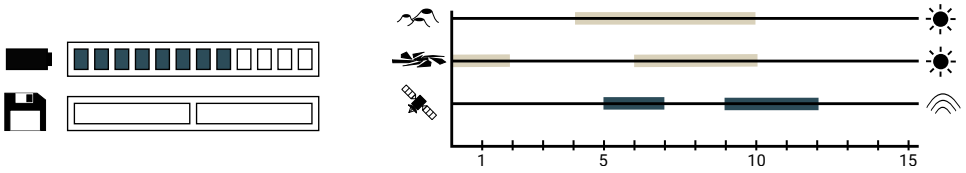


Figure 2: Constraints in the running example. Left: battery level and memory capacity of the rover; Right: time windows in which targets are illuminated and the upload windows of the relay satellite.

Therefore, given the infeasibility of collecting all data samples, it is necessary to identify a trade-off. An *iterative process* can be used to work out those trade-offs. The interaction between the planning system and users could be as follows:

First a sample plan π_1 is presented, that includes collecting and uploading the soil sample of the rock and the X-ray image of the crater group. This does not cover all data samples. Therefore, a user interested in the image of the newly discovered icy area might ask: “Why is the image of the icy area not taken?”. Taking the image and collecting the soil sample of the rock is not possible. The answer to this question is “Because it is not possible to take the image of the ice area if the rock sample is collected.”. We call such dependencies of the form $A \rightarrow \neg B$ **goal conflicts**. Figure 3 shows some of the conflicts that arise in our example.

Based on the newly gained information, the researchers can now decide whether to include the ice image in the mission and forgo the rock sample or to prioritize the rock sample. Assuming the ice image is deemed more valuable, the next sample plan π_2 includes the remaining x-ray image of the crater group and the ice image. From there, the different groups of researchers involved in the mission can continue to ask questions to better understand the conflicts between the data samples better and to refine their preferences.

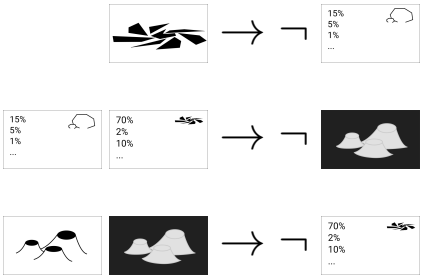


Figure 3: Selection of goal conflicts in the running example. $A \rightarrow \neg B$ means if A is satisfied by a plan then not all goals in B can be satisfied, e. g. for the second conflict, if the soil sample of the rock and the ice surface is taken, then it is not possible to take the x-ray image of the three craters.

If neither group is willing to give in and both data points are to be collected, the question arises “Why is it not possible to collect the rock sample and to take the ice image?”. To address this question, we further consider the constraints of the planning task, in this case: the limited energy of the rover and the upload windows of the relay satellite. By relaxing these constraints, for example by increasing the rover’s initial battery level or the upload window, we can determine the amount of energy and the upload window size necessary to collect both data samples. This results in the answer: “It is possible to collect both data points, if the rover has one additional unit of energy.” Figure 4 shows how the conflict between the ice image and the rock sample weakens as the rover’s battery level increases. While it is not possible to upload both samples with 8 unites of energy, it becomes possible for 9, as long as one can forgo the x-ray images, the crater image and the ice soil sample. Using this information the researchers can determine whether to allocate additional energy to the mission, by for example extending the recharging period or whether this is not feasible, and they have to make a trade-off between the rock sample and the ice image.

So far, the researchers’ preferences only specified which of the data points should be collected. However, their personal preferences could be more complex. For example, they could specify in which order the data points should be collected e. g. “The x-ray image of the crater group should be taken before the soil sample of the ice surface” or express preferences for specific connections e. g. “The rovers should not use the route between the rock and the single crater”. These properties are not reflected in the model because they are not general restrictions of the domain or instance, but rather properties based on personal user preferences and experience. The researchers can add individual user preferences to further shape the sample plans to converge on a solution that not only includes the preferred data points but also collects them according to the researchers’ preferences. Incorporating more complex properties extends the explanations from conflicts between data points to detailed explanations of user preferences.

This example illustrates that even with small instances, the dependencies between goals and user preferences can be difficult to understand, but are crucial to finding a satisfactory compromise.

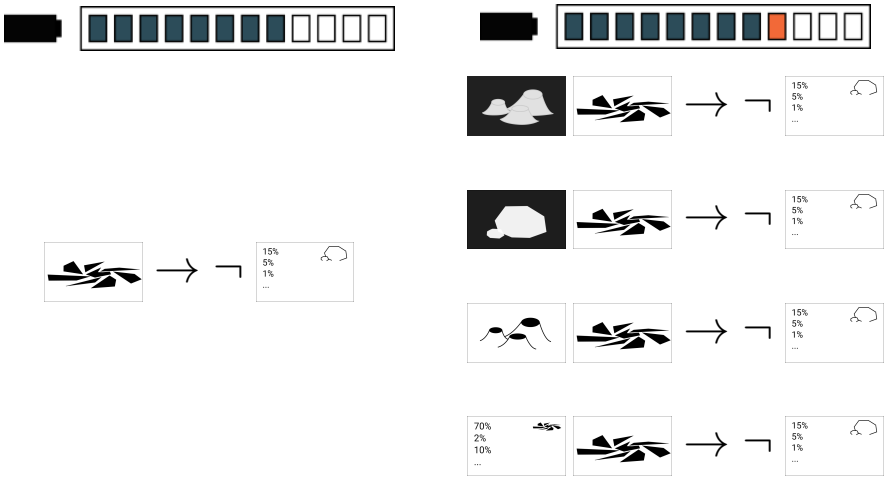


Figure 4: The development of the conflict between the ice image and the rock sample when the battery level is increased. When the battery is increased by one unit, it becomes possible to take the ice image and collect the rock sample, if one is willing to forgo the X-ray images the caters image and the ice sample.

1.2 CONTRIBUTIONS

Chapter 4 We present a formal framework for explanations based on goal conflicts in oversubscription planning that are suitable for an iterative planning process. These explanations use minimal goal conflicts of the form $Q \rightarrow \neg A_1 \cdots Q \rightarrow \neg A_n)$ to address user questions of the form "Why is Q not satisfied by sample plan π?" with "Because then you have to forgo at least one goal in each $A_1, \cdots A_n$." To address the problem (AllMUGS) of computing all minimal unsolvable goal subsets (MUGS), providing the necessary conflict information, we introduce two approaches based on either a goal space search or an exhaustive state space search. For the former, we explore explicit as well as symbolic search to prove individual goal subsets solvable. In the latter approach, which involves tracking the solvable goal subsets in a branch-and-bound search, we introduce heuristics that under-approximate the reachable goal subsets, thereby pruning states that cannot further improve the solvable goal subsets. To handle more complex preferences, we adapt previous work on compilation of temporal preferences specified in finite linear temporal logic (LTL_f). We evaluate the computational feasibility of AllMUGS in an empirical analysis. To evaluate the usefulness of the proposed explanations we have developed an online iterative planning tool (IPEXCO) and conducted an extensive user study.

Chapter 5 To answer follow-up questions in the iterative planning process of the form "Why are the goals in $Q \cup A_i$ in conflict?", we introduce explanations based on relaxations of the planning task. Utilizing minimal relaxations that resolve a conflict, we provide explanations of the form "To resolve $Q \cup A_i$, it is necessary to relax the task to either x or y." Given a set of possible relaxations we address the problem of computing the MUGS for all relaxations (AllRelaxMUGS). We introduce two approaches, exploiting the fact that solvable goal subsets and reachable states, can be propagated from a less to more a relaxed task. The relaxation

explanations approach is instantiated with relaxations based on resource and time window constraints. Using special soft goals reflecting such constraint relaxations, we provide a compilation from AllRelaxMUGS to AllMUGS. An empirical evaluation is performed to assess the feasibility and suitability of each approach for resource and time window constraint relaxations.

Chapter 6 Including additional human preferences not reflected by the task in the conflict analysis allows to tailor the plan and the explanations to the users' preferences. However, as it is often difficult for human users to formalize their preferences, we instead propose an approach that allows them to rate sample plans as good or bad in relation to their preferences. This allows to extract the preference by exploiting previous work on LTL_f formula learning from sample traces. We empirically evaluate the approach of learning temporal preferences from annotated sample plans. We assess different approaches to generating sample plans that are commonly used in diverse planning and compare the accuracy of the learned preference with the target preference as a function of the formula size and the number of plans.

1.3 PUBLICATIONS

The thesis is mainly based on the following publications.

Conference Publications

- **Rebecca Eifler, Jeremy Frank and Jörg Hoffmann**
  Explaining Soft-Goal Conflicts through Constraint Relaxations.
  Proceedings of the 31th International Joint Conference on Artificial Intelligence (2022)
- **Rebecca Eifler, Martim Brandao, Amanda Coles, Jeremy Frank and Jörg Hoffmann**
  Evaluating Plan-Property Dependencies: A Web-Based Platform and User Study.
  Proceedings of the 32th International Conference on Automated Planning and Scheduling (2022)
- **Valentin Seimetz, Rebecca Eifler and Jörg Hoffmann**
  Learning Temporal Plan Preferences from Examples: An Empirical Study.
  Proceedings of the 30th International Joint Conference on Artificial Intelligence (2021)
- **Rebecca Eifler, Marcel Steinmetz, Alvaro Torralba and Jörg Hoffmann**
  Plan-Space Explanation via Plan-Property Dependencies: Faster Algorithms & More Powerful Properties.
  Proceedings of the 29th International Joint Conference on Artificial Intelligence (2020)
- **Rebecca Eifler, Michael Cashmore, Jörg Hoffmann, Daniele Magazzeni and Marcel Steinmetz**
  A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning.
  Proceedings of the 34th AAAI Conference on Artificial Intelligence (2020)

System Demonstration

- **Rebecca Eifler and Jörg Hoffmann**
  A Web-Based Platform for Iterative Planning with Plan Explanations.
  System Demonstration at the 30th International Conference on Automated Planning and Scheduling (2020)

Additionally, the following papers have been published during the authors doctoral studies but are not part of this thesis.

Conference Publications

- **Rebecca Eifler, Maximilian Fickert, Jörg Hoffmann and Wheeler Ruml**
  Refining Abstraction Heuristics during Real-Time Planning.
  Proceedings of the 33th AAAI Conference on Artificial Intelligence (2019)

- **Rebecca Eifler and Maximilian Fickert**
  Online Refinement of Cartesian Abstraction Heuristics.
  Proceedings of the 11th Annual Symposium on Combinatorial Search (2018)

Workshop Publications

- **Aleena Siji, Rebecca Eifler, Daniel Fišer, and Jörg Hoffmann**
  Action Policy Explanations in Oversubscription Planning.
  Proceedings of the International Workshop of Human-Aware and Explainable Planning (2023)

CHAPTER 2

RELATED WORK EXPLAINABLE AI

Much thought has been given to explainable AI in recent years, as shown by the numerous projects focused on this topic. The XAI program of the Defense Advanced Research Projects Agency (DARPA) [Gunning and Aha, 2019] is among the first of many projects related to safe, robust and trustworthy AI funded by the EU's research and innovation funding program [Horizon Europe, 2021]. There are many approaches to explainable AI, which differ in terms of why an explanation is needed and how it can be provided. In the following, we first provide a general overview of the challenges of explainable AI (XAI) and then focus on explainable AI planning (XAIP) in particular.

2.1 EXPLAINABLE ARTIFICIAL INTELLIGENCE

AI systems are used in a wide variety of areas, sometimes replacing humans completely or meant to supporting them in a task. This includes tasks like image classification, speech recognition, content recommendation, satellite spacecraft operations planning [Pralet et al., 2022, Smith et al., 1999] and flight planning [Geißer et al., 2020]. Depending on the area the AI system is applied to, for example medicine [de Vries et al., 2023], finance [Weber et al., 2024, Pozanco et al., 2023] or law enforcement [Matulionyte and Hanif, 2021, Khan et al., 2023], decisions can have crucial effects on humans and thus require explanations. But even in non-live-changing scenarios, understanding the decisions that a black-box system makes is essential for a satisfactory and productive interaction with a human.

Explanations can address different objectives [Hoffman et al., 2018, Langer et al., 2021, Chakraborti et al., 2019a] such as *fairness*, *trust* and *performance*, which go beyond *understanding* the result of an AI system. The European Parliament decided, a user has the "Right to Explanations" [Goodman and Flaxman, 2017], when affected by algorithmic decisions making. Their major concern is *discrimination*. While, they do not specify what they exactly mean by "Explanation", their objective for explanations is to ensure *fairness* [Mehrabi et al., 2021]. To *trust* someone or a system, without the possibility to confirm the correctness of their decisions is not satisfactory. Explanations can help the user to explore the boundaries of the capabilities of the AI to decide for which tasks they can rely on the AI [Hoffman et al., 2018, 2021]. Last but not least, explanations can also affect the collaborative performance of the human and the AI system [Hoffman et al., 2018]. This does

not only include performance with respect to the primary goal of the task, but also the user's performance of predicting the AI's decision.

The main objective of our explanations is understanding, more precisely the understanding of the dependencies/conflicts between different goals/preferences. Especially, expert users have specific expectations or preferences on how things should be done and if these are not satisfied, explanations are necessary to understand why [Smith, 2012, Krarup et al., 2021]. In addition, the aim is to improve performance, i. e. to reach an agreement more quickly on a plan that satisfies all involved parties.

To address any of these objectives there are many approaches which can be classified with respect to different properties. For a more exhaustive overview we refer to Speith [2022] for general XAI and for XAI in planning to Chakraborti et al. [2020].

There are two main conceptual ideas on how to allow a user to follow a decision made by an AI system. One can either use a *transparent model* or provide *post-hoc explanations* [Speith, 2022, Arrieta et al., 2020]. A model is transparent, if it does not require any additional information for a user to understand the decision-making process, but itself is understandable. Models considered transparent are for example, decision trees or rule based learners [Arrieta et al., 2020]. However, transparency is only given up to a certain model size. Even for the simple model semantics, at some point the size and as a result increasing complexity, makes additional explanations necessary [Langer et al., 2021]. Post-hoc explanations, provide additional information to either explain the whole model (global explanation) or to address specific user questions (local explanations) [Speith, 2022].

Planning itself, could be considered transparent, however the size of the models we address are too large to be understandable by a human. Thus, additional explanations are necessary. Our approach can theoretically provide global explanations, to the granularity chosen by the user. However, within the iterative planning framework we focus on, local explanations addressing specific user questions of the form "Why does the sample plan not satisfy Q?".

This type of question is called *contrastive*. The user presents a fact, here Q, they want to or expect to be satisfied and asks why this is not the case. Users either explicitly state a contrasting fact, i. e. "Why does the sample plan satisfy P rather than Q?" or assume that its clear from context, e. g. "Why do you turn right? (rather than left)". Findings from social sciences [Miller, 2019, 2021] and a user study [Krarup et al., 2021] show that contrastive questions and contrastive explanations are one of the most common ways in which humans ask for and give explanations. Thus, we build on initial ideas by Hoffmann and Magazzeni [2019] to provide contrastive explanations in AI Planning.

In the context of post-hoc explanations the question arises when to provide explanations. Most reactive explanation systems, responding to a user request, focus on providing explanations during or after the generation of a solution. Proactive explanation systems on the other hand often focus on providing explanations during the execution of a solution. The decision to provide explanations is influenced by two factors: the collaborative settings, i. e. the extent to which the human is only an observer or whether the agent and the human collaborate on a task, and the shared knowledge of the environment, capabilities and goals. Goal and plan recognition [Meneguzzi and Pereira, 2021] is therefore crucial for deciding

when and whether to give explanations. Caglar and Sreedharan [2024] address a setting where a human is supported by an agent. If the human performs an action that is likely to result in a failure state, the agent intervenes by providing the necessary information to help the human understand the consequences of their action. In robot systems, studies have shown that proactive explanations of failure leads to greater trust in the system [LeMasurier et al., 2024]. Jain et al. [2023] point out that the initial decision-making process of the agent should be explained to allow the human to form a correct mental model of the agent. Later on, explanations should be reserved to decisions that deviate from the human's expectation based on the acquired model.

Once decided when, providing explanations to a human user is a two-step process [Miller, 2019].

1) The causal information for the explanations needs to be determined.

2) The part relevant to the user is communicated appropriately.

In the work presented here, we focus mainly on the first step and leave the selection of the "best" explanation for a particular user and the process of communicating this explanation to future work. References to work that focuses on the human interaction part is given at the end of the next section.

2.2 EXPLAINABLE AI PLANNING

In contrast to Machine Learning, model based techniques are better suited for explanations. The explicit representation of world knowledge and the reasoning approaches facilitate access to causal information. The recent interest in explainable AI Planning (XAIP) lead to many approaches covering different planning formalism, explanation approaches and use cases. The workshop for Human aware Explainable Planning (HAXP[1]) at the international conference on Automated Planning and Scheduling (ICAPS) reflects the large interest in explanations in the planning community.

In the following, we do not consider explanations of the reasoning approaches themselves, i. e. how a specific planning algorithm works (overview Baier and Kaisers [2020], MCTS [An et al., 2024]), but rather on explanations of the resulting plan or possible plan space based on the given model. Explanations are necessary if the plan does not align with the users expectations or does not satisfy the user because it does not fulfill certain properties desired by the user, or the user simply does not understand the internal process of the plan. These issues arise due to the user's limited reasoning abilities and/or because the mental model of the human does not align with the planning agent's model.

Misaligned Human and Planning Agent's Models To resolve the misalignment of the human and agent's model, explanations based on model reconciliation [Chakraborti et al.,

[1] www.haxp.org

2017, Sreedharan et al., 2021] have been introduced. Assuming that the agent knows the human model, the goal is to identify the model differences and update the human model to match the proposed plan. This approach has been extended in many directions. To weaken the assumption about the knowledge of the human model the approach has been extended to handle uncertainty in the human model [Sreedharan et al., 2018a] and introduced the possibility to learn suitable explanations based on the applicability of sample plans in the humans model [Sreedharan et al., 2019a]. Instead of providing the model update in post-hoc explanations, the model updates can also be provided within the plan [Sreedharan et al., 2017, 2020a]. This results in self-explaining plans that incorporate communication actions to update the human's model directly or task actions with epistemic effects indirectly updating the model. In this context also the trade-off between an explicable plan, following the users' expectation, and explanations is addressed [Sreedharan et al., 2017, 2024]. In addition, safety should not be sacrificed for explicability [Hanni et al., 2024]. In case of large model discrepancies one global model update is not suitable. Local updates that are based on a dialogue between system and human allow to share beliefs as well as to request information [Vasileiou et al., 2023]. These approaches have been applied to different use cases in different tools, for example in RADAR-X [Grover et al., 2020, Karthik et al., 2021] a decision support system for rescue missions. While the original definition [Sreedharan et al., 2021] is defined as the reconciliation of a model represented by a planning task, this approach is also implemented in a logic based framework [Vasileiou et al., 2019].

Limited Reasoning Capabilities If the models of the human and the system align, explanations may still be required due to the human's limited reasoning capabilities.

Before we get to post-hoc explanations, we first consider approaches that provide *transparent* plans such that the plan is self-explanatory. Lindsay [2021] argues that the representation of the planning strategies as rule based policies results in a transparent planning approach. Based on learned features of explainable and predictable plans, Zhang et al. [2017] generate and select plans with these features. MacNally et al. [2018] state that transparent plans should be designed such that the goal objective is clear as soon as possible. Transparency often comes with an overhead, whether due to unnecessary explanations or suboptimal plans, which can be a crucial factor depending on the application. The trade-off between explanations embedded into the plan and post-hoc explanations is investigated by Lindsay et al. [2020].

Next we consider post-hoc explanations. We distinguish the following categories of user questions:

- "Why is A in plan π?"
- "Why is A in plan π rather than B?"
- "Why is there no plan?"

To address the question "Why is A in plan π?", causal chain explanations [Seegebarth et al., 2012] argue via the causal links between an action with fact p as a precondition

and an action with p as effect. Causal link explanations have also been extended to fully observable nondeterministic planning [Sreedharan et al., 2022] and state action policies [Sreedharan et al., 2023]. Sohrabi et al. [2011] do not answer the question "Why is A in plan π?" but rather the related questions "How happens A?" from a partially defined state. As an answer, a set of facts that need to hold initially and a plan π satisfying A is given.

By far the most interest has been shown in contrastive explanations addressing the question "Why is A in plan π rather than B?". Krarup et al. [2021] and Murray et al. [2022] provide a whole framework for contrastive explanations using the differences between the original plan π satisfying A and an alternative plan π' that satisfies B. If the alternative plan is of lower quality, Coles and Krarup [2024] provide explanations based on abstractions to identify the part of the model causing the quality difference. These explanations and the model refinements to provide alternative plans are designed for the iterative planning approach described by Smith [2012]. The approach by Lindsay and Petrick [2021a] uses abstract explanations by focusing on individual objects, such as the rover and its movement or in a transportation domain the location changes of a single package. Explanations are provided based on alternative transition sequences for individual objects as well as a causal justification for the chosen sequence. Kim et al. [2019] assume that the contrastive question is not formulated as two contrastive properties of a given plan, but as two sets of contrasting sample plans. As an explanation they use temporal properties that the positive samples have in common, while they are not satisfied by the negative samples.

Answering the question "Why is there no plan?", i. e. explanation of unsolvability, may be necessary if, for example, the model is faulty or the user has imposed restrictions that make the model unsolvable. Göbelbecker et al. [2010] had the former reason in mind, when introducing excuses for unsolvability. Excuses are minimal changes to the initial state in the form of changes to the state of an object or additional objects. In contrast, Sreedharan et al. [2019b] use solvable abstractions and landmarks thereof as explanations.

To make most of these generic approaches applicable to specific domains, some domain specific specifications are necessary. Lindsay [2019, 2020] provides more tailored explanations without domain dependent implementations based on problem structures, like moving or transporting, common to many domains.

Explanations of Policies In probabilistic planning, a solution is generated by executing a policy, a function that maps states to actions. As before if this solution, i. e. the decisions of the policy in a given state or over a sequence of states, does not align with the users' expectations, an explanation is required. In a multi-objective setting, Sukkerd et al. [2020] facilitates finding a trade-off with explanations based on the Pareto-optimal alternative policies. Sreedharan et al. [2023] extend causal link explanations to policies. Sreedharan et al. [2025] provide contrastive explanations for alternative policies suggested by the user. Model simplifications, such as abstractions, determinization and decomposition, are used to create explanations, which are simplified models in which the original policy performs *better*,

with respect to plan cost and probability of achieving the goal, than any of the alternative policies.

Plan generation based on learned policies [Toyer et al., 2020, Ståhlberg et al., 2022], for probabilistic but also classical planning, is now more widely used. A first approach extends abductive explanations from explainable deep learning to policies [Selvey et al., 2023]. It explains the execution sequence of a deterministic state-action policy by identifying the properties of a state that ensures the specific execution.

Use Cases The introduced approaches have been applied to different use cases, ranging from teaching children [Tulli et al., 2020] to set up a home entertainment system [Seegebarth et al., 2012]. Tulli et al. [2020] teach children a game using contrastive explanations to facilitate learning. They use the difference between an optimal plan and different suboptimal plans as explanation. The causal link explanations by Seegebarth et al. [2012] are applied by Bercher et al. [2014] to the task of setting up a home entertainment system in case where replanning is necessary due to unexpected problems. Brandao et al. [2021a] introduced contrastive explanations to path planning which was then extended by Alsheeb and Brandao [2023] to road navigation systems to provide contrastive explanations based on properties of the map and road conditions. To support the process of model acquisition for a dialog system, Sreedharan et al. [2020b] provide explanations in case of an unsatisfiable specification or unexpected behavior based on model reconciliation [Sreedharan et al., 2019b, 2021]. Lindsay and Petrick [2021b] apply explanations to a tour guide system which interacts with a user through a dialogue system. They provide explanations on two abstraction levels, first explaining the next high level task, like visiting a café, and then, if necessary, explanations following the model reconciliation approach to align the systems and human's model of a given map for a successful guidance of the human.

Explanation Selection and Communication So far, all approaches mainly address the first step of explanations, the generation of the causality information. However, the second step, i.e. the selection of the most relevant explanation and the communication to the user is equally important. One of the aspects to be considered here is to choose the right level of abstraction that corresponds to the knowledge of the users. Within the model reconciliation framework Sreedharan et al. [2018b] use the minimal state abstractions of the systems model that addresses the conflict with the user's contrastive suggestion. Vasileiou and Yeoh [2023] instead opt for a task-specific vocabulary the human is familiar with to provide personalized explanations in the logic based model reconciliation approach [Vasileiou et al., 2019]. However, not only choosing the best suited explanation, but also the way it is communicated to the user has an impact on the clarity of the explanation. Kumar et al. [2022] chose visualizations as a means of communication, to simplify the understanding of model reconciliation explanations introduced by Sreedharan et al. [2021].

The recent boom in large language models (LLMs) has prompted a discussion on their application in generating and communicating explanations. While the capabilities of LLMs in reasoning and thus planning are limited [Valmeekam et al., 2024], there is the proposal for leveraging them in conjunction with a verifier to ensure the correctness of the generated plan [Kambhampati et al., 2024]. A similar approach can be considered for explanations,

where for example LLMs provide a natural language interface to logic-based explanations or where the explanations provided by an LLM are checked and refined in an iterative process with an explanation framework that can ensure correctness.

CHAPTER 3

BACKGROUND

We next introduce the background on classical and oversubscription planning and heuristic state space search. This is followed by a short introduction to linear temporal logic.

3.1 PLANNING AND SEARCH TECHNIQUES

First we introduce the formalism for classical and oversubscription planning and the resulting search spaces represented by a labeled transition system.

3.1.1 OVERSUBSCRIPTION PLANNING

There are different variants of planning, distinguished by their expressiveness. The variants include probabilistic action outcomes, temporal aspects and continuous variables. For an overview we refer the reader to [Ghallab et al., 2004]. In this work, we focus on classical planning[1]. A classical planning task consists of states that are described by discrete variables with finite domains, all of which are known in advance in their entirety. The actions to change these states are deterministic, and their effect is instant. As the formalism for classical planning we use the *finite-domain representation (FDR)* [Bäckström and Nebel, 1995, Helmert, 2009].

DEFINITION 1: CLASSICAL PLANNING TASK

An FDR planning task is a tuple $\tau = (V, A, c, I, G)$, where

- V is a finite set of **variables**, each $v \in V$ being associated with its finite **domain** D_v. A complete assignment to V is called a state.
- A is a finite set of **actions**. Each action $a \in A$ has a **precondition** pre_a and an **effect** eff_a, both partial assignments to V.
- $c : A \to \mathbb{R}_0^+$ is the action **cost** function.
- I is the **initial state**.

[1] In Section 5.4 we introduce planning with resource and time window constraints. However, these extensions can be compiled to classical planning and do not add any expressiveness.

- G is the **goal**, defined by a partial state.

The partial FDR model of our running example, not taking any constraints into account, is given in Example 1.

EXAMPLE 1: FDR PLANNING TASK

- Variables V:
  - The location a rover can be at: R with $D_R = \{\texttt{crater1}, \texttt{rock}, \texttt{crater3}, \texttt{ice}\}$
  - A data sample a rover can take: $dp \in \{\texttt{image-ice}, \texttt{image-crater3}, \texttt{x-ray-image-crater3}, \texttt{x-ray-image-rock}, \texttt{soil-sample-rock}, \texttt{soil-sample-ice}\} = DP$ with $D_{dp} = \{\texttt{todo}, \texttt{in-memory}, \texttt{uploaded}\}$
- Actions A:
  - move rover from location l_1 to location l_2: $\texttt{drive(l1,l2)}$ for $\{l_1, l_2\} \in \{\{\texttt{crater1}, \texttt{rock}\}, \{\texttt{rock}, \texttt{crater3}\}, \{\texttt{crater3}, \texttt{ice}\}, \{\texttt{ice}, \texttt{crater1}\}\}$
    - $pre : \{R = l_1\}$
    - $\mathit{eff} : \{R = l_2\}$
  - take a soil sample at location l: $\texttt{soil-sample(l)}$ for $l \in \{\texttt{rock}, \texttt{ice}\}$
    - $pre : \{R = l, \texttt{soil-sample-}l = \texttt{todo}\}$
    - $\mathit{eff} : \{\texttt{soil-sample-}l = \texttt{in-memory}\}$
  - take an image at location l: $\texttt{image(l)}$ for $l \in \{\texttt{crater3}, \texttt{ice}\}$
  - take an x-ray image at location l: $\texttt{x-ray-image(l)}$ for $l \in \{\texttt{crater3}, \texttt{rock}\}$
  - upload data point d to relay satellite: $\texttt{upload(d)}$ for $d \in DP$
- Action Cost:
  - $c(\texttt{drive(l1,l2)}) = v$ for $(\{l1, l2\}, v) \in \{(\{\texttt{crater1}, \texttt{rock}\}, 3), (\{\texttt{rock}, \texttt{crater3}\}, 1), (\{\texttt{crater3}, \texttt{ice}\}, 3), (\{\texttt{ice}, \texttt{crater1}\}, 2)\}$
  - $c(\texttt{soil-sample}(l)) = 1$ for $l \in \{\texttt{rock}, \texttt{ice}\}$
  - $c(\texttt{image}(l)) = 1$ for $l \in \{\texttt{crater3}, \texttt{ice}\}$
  - $c(\texttt{x-ray-image}(l)) = 2$ for $l \in \{\texttt{crater3}, \texttt{rock}\}$
  - $c(\texttt{upload}(d)) = 1$ for $d \in DP$
- Initial State: $\{R = \texttt{crater1}\} \cup \{dp = \texttt{todo} \mid dp \in DP\}$
- Goal: $\{x = \texttt{uploaded} \mid x \in DP\}$

We will refer to variable-value pairs $v = d$ with $v \in V$ and $d \in D_v$ as **facts** and identify partial variable assignments with sets of facts. The value assigned to a variable v in state s is referred to by $s(v)$. An action a is **applicable** in a state s if $pre_a \subseteq s$. The set of all applicable actions in state s is defined as $A[s] := \{a \mid a \in A, pre_a \subseteq s\}$. Applying a in s leads to successor state $s[\![a]\!]$, that is defined as

$$s[\![a]\!](v) = \begin{cases} \mathit{eff}_a(v) & \text{if } \mathit{eff}_a(v) \text{ defined} \\ s(v) & \text{otherwise} \end{cases}$$

The outcome state of an iteratively applicable action sequence π is denoted by $s[\![\pi]\!]$. The prefix of an action sequence $\pi = a_0, \cdots, a_{i-1}, a_i, a_{i+1}, \cdots, a_n$ is defined as *prefix*$(\pi, i) = a_0, \cdots, a_{i-1}$.

The state space of such an FDR task is defined by a labeled transition system.

DEFINITION 2: LABELED TRANSITION SYSTEM

A labeled transition system (LTS) is a tuple $\Theta = (\mathcal{S}, L, c, T, I, \mathcal{S}_G, b)$, where

- $\mathcal{S}$ is a finite set of **states**,
- L is a finite set of **transition labels**,
- $c : L \mapsto \mathbb{R}_0^+$ is the **cost function** mapping each label to a non-negative cost,
- $T \subseteq \mathcal{S} \times L \times \mathcal{S}$ is a set of **transitions**,
- $I \in \mathcal{S}$ is the **initial state**,
- $\mathcal{S}_G \subseteq \mathcal{S}$ is the set of **goal states**
- $b \in \mathbb{R}_0^+ \cup \{\infty\}$.

A path in a LTS is a sequence of labels $\pi = l_0, \ldots, l_n$ such that there exists a sequence of states $s_0, \ldots, s_n$ with $(s_i, l_i, s_{i+1}) \in T$. The cost of a path π is the sum of the costs of its labels, cost$(\pi) = \sum_{i=0}^{n} c(l_i)$. A path is a **solution** for state s if $s_0 = s$ and $s_n \in \mathcal{S}_G$ and cost$(\pi) \leq b$. A solution for I is a solution for the LTS. A state s is **reachable from state** s' if there is a path π from s' to s and cost$(\pi) \leq b$. A state s is **reachable** in an LTS if it is reachable from I. The set of all reachable states in an LTS is denoted by $\mathcal{S}^r$ and the set of all reachable goal states $\mathcal{S}^r \cap \mathcal{S}_G$ by $\mathcal{S}_G^r$. We consider **deterministic** LTSs, meaning for all $s \in \mathcal{S}$ and $l \in L$ there is at most one $s' \in \mathcal{S}$, such that $(s, l, s') \in T$.

The state space of a classical planning task is defined by the following LTS.

DEFINITION 3: STATE SPACE

The **state space** of a planning task $\tau = (V, A, c_\tau, I_\tau, G)$ is the LTS $\Theta_\tau = (\mathcal{S}, L, c_\Theta, T, I_\Theta, \mathcal{S}_G, b)$ where

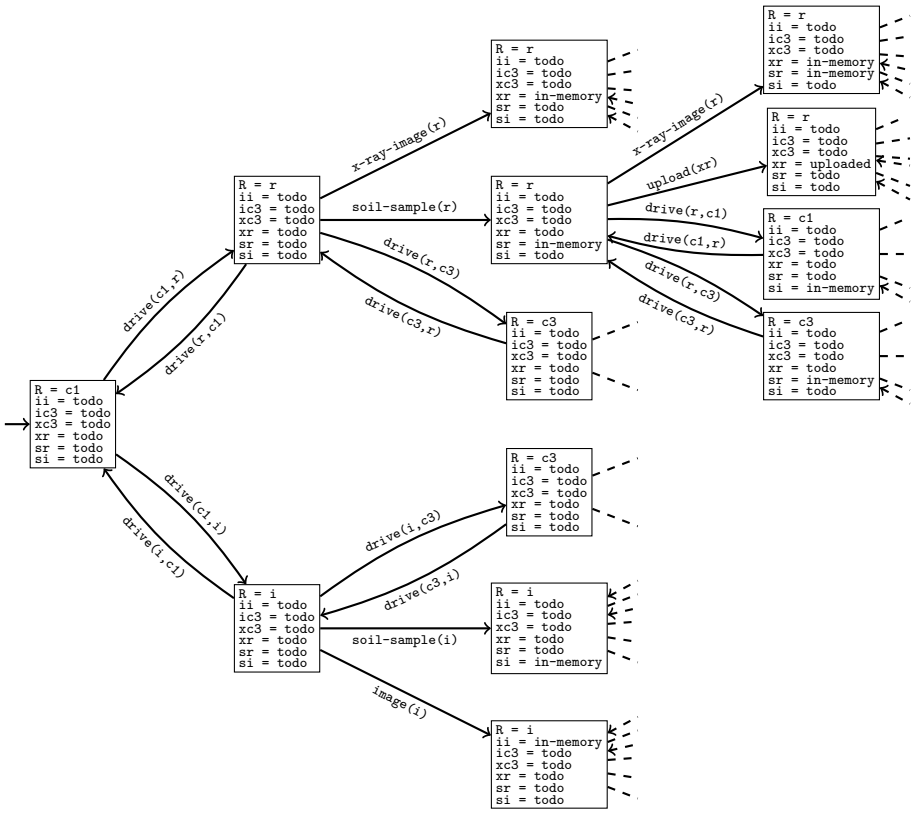
- $\mathcal{S}$ is the set of all states of τ,

- $L = A$, the labels are the actions of τ,
- the cost function $c_\Theta = c_\tau$ is the cost function of τ,
- $(s, a, s') \in T$, iff $s, s' \in S$, $a \in A$, $pre_a \subseteq s$ and $s[\![a]\!] = s'$,
- the initial state $I_\Theta = I_\tau$ is the initial state of τ,
- $\mathcal{S}_G = \{s \in S \mid G \subseteq s\}$,
- $b = \infty$.

An exemplary part of the state space based on the FDR definition of the running example in Example 1 is given in Example 2.

EXAMPLE 2: STATE SPACE

Below a part of the state space of the FDR task in Example 1 is depicted. We use the following abbreviations: `crater1`: `c1`, `rock`: `R`, `crater3`: `c3`, `ice`: `i` and `R` for the rover.



A solution π for state s in state space Θ_τ is a **plan** for s in task τ. We say that task τ is **solvable** if a plan for the initial state exists, and **unsolvable** otherwise. A plan π is **optimal** if its cost $\text{cost}(\pi)$ is minimal among all plans.

In the following we consider **oversubscription** planning [Smith, 2004, Mirkis and Domsh-

lak, 2013, Domshlak and Mirkis, 2015, Aghighi and Jonsson, 2014] an extension of classical planning with an overall cost bound for the plan cost. Additionally, we distinguish two types of goals.

DEFINITION 4: OSP TASK

An **oversubscription planning (OSP) task** is a tuple $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ like an FDR task but with

- **hard** goal G^{hard},
- **soft** goal G^{soft},
- **cost bound** $b \in \mathbb{R}_0^+$,

  where G^{hard} and G^{soft} are partial states on disjoint sets of variables.

The hard goal is defined as the set of goals, that must be satisfied by a solution for a given task. In contrast, the soft goal represents the additional goals, that we would like to be satisfied. However, due to the cost bound this might not be possible. Example 3 describes a possible OSP version of our running example.

EXAMPLE 3: OSP TASK

In an OSP version of our running example, one could for example consider uploading the image of the ice surface as mandatory and thus make it a hard goal, while all other data points are optional and thus soft goals. A cost bound of 15, reflecting the available energy, prevents all data points to be uploaded.

- hard goal: $\{\texttt{image-ice} = \texttt{uploaded}\}$
- soft goal: $\{dp = \texttt{uploaded} \mid dp \in DP\} \setminus \{\texttt{image-ice}\}$
- cost bound: $b = 15$

A possible plan π including the hard goal `image-ice`= `uploaded` and the two soft goals `x-ray-image-rock` = `uploaded` and `image-crater3` = `uploaded`:

- `drive`(`crater1`,`rock`) (3)
- `x-ray-image`(rock) (2)
- `drive`(`rock`, `crater3`) (1)
- `image`(`crater3`) (1)
- `drive`(`crater3`,`ice`) (3)
- `image`(`ice`) (1)
- `upload`(`x-ray-image-rock`) (1)
- `upload`(`image-ice`) (1)
- `upload`(`image-crater3`) (1)

> The numbers in the brackets indicate the action cost, the plans cost cost$(\pi) = 14$. With a cost bound of 15 no additional soft goal can be achieved.

The state space of an OSP task is analogous to that of a classical planning task, with the exception of the definition of goal states and the finite cost bound.

DEFINITION 5: STATE SPACE OSP TASK

The **state space** of an OSP task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ is the LTS $\Theta_\tau = (\mathcal{S}, L, c_\Theta, T, I_\Theta, \mathcal{S}_G, b_\Theta)$ where $\mathcal{S}$, L, c_Θ, T and I_Θ are defined as for a classical planning task, and where $\mathcal{S}_G = \{s \in S \mid G^{\text{hard}} \subseteq s\}$ and $b_\Theta = b$.

We say that $G \subseteq G^{\text{soft}}$ is **solvable in** τ if $\tau' = (V, A, c, I, G^{\text{hard}} \cup G, \emptyset, b)$ is solvable.

Usually, the goal facts in G^{soft} are assigned a **utility** $u : G^{\text{soft}} \mapsto \mathbb{N}_0$. The utility of a plan π is then given by the utility of the final state $u(\pi) = \sum_{g \in (G^{\text{soft}} \cap I[\![\pi]\!])} u(g)$. A plan π is **utility-optimal**, if there is no plan π' where $u(\pi') > u(\pi)$. Here, however, we assume that the utility function is not given because the user's preferences with regard to the soft goals are difficult to specify or have not yet been formed.

3.1.2 HEURISTIC STATE SPACE SEARCH

One of the most effective approaches for solving classical planning and OSP tasks is heuristic state space search [Bonet and Geffner, 2001]. The state space is systematically explored, typically starting from the initial state I in order to then generate the successor states by applying actions. To decide which state to expand next, the states are ranked by a **heuristic**, which is a function estimating the plan cost from a state to a goal state.

DEFINITION 6: CLASSICAL PLANNING HEURISTIC

A **heuristic** $h : \mathcal{S} \mapsto \mathbb{R}_0^+ \cup \infty$ is a function that maps each state s of a planning task τ to an estimate of the cost of a plan from s, or to ∞ to indicate that there is no plan.
The **perfect heuristic** h^* maps each state to the cost of an optimal plan, or to ∞ if no plan exists.
A heuristic h is **admissible** if for all states s, $h(s) \leq h^*(s)$.

Heuristics are used to guide the search towards a goal. For optimal planning A^* [Hart et al., 1968] is the most commonly used search algorithm (see Algorithm 1). It explores states in the order of $g(s) + h(s)$, where g is the cost to reach state s and h the estimation of reaching a goal state from s. A^* requires an admissible heuristic to produce optimal plans. For satisficing planning *greedy best-first search* (GBFS) [Doran and Michie, 1966] (see Algorithm 1) can be used. It greedily follows the most promising state determined by the heuristic.

Algorithm 1 A^* and Greedy Best First Search (GBFS)

Given: task τ, heuristic h
function A^*/GBFS(τ, h)
  $g(s) \leftarrow \infty$ for all states $s \neq I$ and $g(I) \leftarrow 0$
  $f(I) \leftarrow h(I)$
  $\mathcal{O} \leftarrow \{(I, 0)\}$ ▷ *priority queue ordered by ascending* $f(s)$
  while $|\mathcal{O}| > 0$ **do**
    $s, g_s \leftarrow$ POP($\mathcal{O}$)
    if $g(s) < g_s$ **then** ▷ *duplicate check*
      continue
    if $G \subseteq s$ **then**
      return SOLUTION(s)
    for all $a \in A(s)$ **do**
      $s' \leftarrow s[\![a]\!]$
      $g_{s'} \leftarrow g(s) + c(a)$
      if $g_{s'} < g(s')$ **then** ▷ *duplicate check*
        $g(s') \leftarrow g_{s'}$
        $f(s') \leftarrow g(s') + h(s')$ for A^* and $f(s') \leftarrow h(s')$ for GBFS
        $\mathcal{O} \leftarrow \mathcal{O} \cup (s', g(s'))$
  return failure

To efficiently compute our explanations, we will build on existing heuristics.

Max Heuristic The max heuristic [Bonet and Geffner, 2001] estimates the remaining cost to achieve all goals G by the cost to achieve the most expensive $g \in G$.

DEFINITION 7: MAX HEURISTIC

Let $\tau = (V, A, c, I, G)$ be a planning task. The **max heuristic** $h^{\max}$ for τ is the function $h^{\max}(s) := h^{\max}(s, G)$ where $h^{\max}(s, G')$ is the point-wise greatest function that satisfies

$$h^{\max}(s, G') = \begin{cases} 0 & G' \subseteq s \\ \min_{a \in A, g' \in eff_a} c(a) + h^{\max}(s, pre_a) & G' = \{g'\} \\ \max_{g' \in G'} h^{\max}(s, \{g'\}) & |G'| > 1 \end{cases}$$

$h^{\max}$ is admissible. For further information we refer to [Bonet and Geffner, 2001].

Abstraction Heuristic In optimal planning a commonly used family of admissible heuristics are abstraction heuristics [Helmert et al., 2007]. Abstractions group states of a transition system into abstract states.

DEFINITION 8: ABSTRACTION

Let $\Theta = (\mathcal{S}, L, c, T, I, \mathcal{S}_G)$ be a transition system. An **abstraction** of Θ is a surjective function $\alpha : S \mapsto S^\alpha$. The **abstract state space** induced by α is the transition system $\Theta^\alpha = (\mathcal{S}^\alpha, L^\alpha, c^\alpha, T^\alpha, I^\alpha, \mathcal{S}_G{}^\alpha)$ defined by:

- $I^\alpha = \alpha(I)$
- $S^{\alpha G} = \{\alpha(s) \mid s \in \mathcal{S}_G\}$
- $T^\alpha = \{(\alpha(s), l, \alpha(t) \mid (s, l, t) \in T)\}$

A heuristic based on such an abstraction uses the perfect heuristic in the abstract state space to estimate the remaining cost in the concrete state space.

DEFINITION 9: ABSTRACTION HEURISTIC

Let $\Theta = (\mathcal{S}, L, c, T, I, \mathcal{S}_G)$ be a transition system and α an abstraction of Θ. The **abstratction heuristic** induced by α is the heuristic function $h^\alpha : S \mapsto \mathbb{R}_0^+ \cup \infty$ which maps each state $s \in S$ to $h^*_{\Theta^\alpha}(\alpha(s))$, i. e. to the cost of an optimal plan of $\alpha(s)$ in Θ^α.

Many variants of abstractions and methods for their calculation have been introduced. Abstractions can be based on the projection of individual variables [Culberson and Schaeffer, 1998, Edelkamp, 2001], the abstraction of individual variable domains [Domshlak et al., 2009], the cross product of subsets of domains [Seipp and Helmert, 2013] or any arbitrary mapping [Helmert et al., 2007].

Potential Heuristics Potential heuristics [Pommerening et al., 2015] assign a numeric value to each fact called *potentials*. The resulting heuristic estimate of a state s is the sum of the potentials of all facts satisfied by s.

DEFINITION 10: POTENTIAL HEURISTICS

Let τ be a classical planning task with Variables V and facts $\mathcal{F}$. Given a potential function $pot : \mathcal{F} \mapsto \mathbb{R}$ the potential heuristic for pot of state s is:

$$h^{pot}(s) = \sum_{v \in V} pot(v = s(v))$$

The potentials are chosen such that it is guaranteed that the resulting heuristic is admissible. A linear program ensuring these constraints can optimize with respect to different criteria [Seipp et al., 2015], e.g. the initial state or a sample set of states.

3.2 FINITE LINEAR TEMPORAL LOGIC

A language commonly used to define temporal properties is Linear Temporal Logic (LTL) [Pnueli, 1977]. To define more complex user preferences we use **finite Linear Temporal Logic** (LTL$_f$) [Baier and McIlraith, 2006a, De Giacomo and Vardi, 2013], an adaption of LTL to finite traces.

For a planning task τ with facts $\mathcal{F}$, a LTL$_f$ formula ϕ can be composed as follows:

$$\phi ::= \texttt{final} \mid \texttt{true} \mid \varphi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \phi_1 \,\mathsf{U}\, \phi_2$$
$$\text{with } \varphi \in \mathcal{F}$$

LTL$_f$ formulas are interpreted over a finite sequence of states (finite trace) $\sigma = s_0 s_1 \cdots s_n$ where each state $s_i \subseteq \mathcal{F}$ is a set of facts satisfied by s_i. We use the abbreviation $\sigma[i]$ for $s_i \cdots s_n$. Given a finite trace σ and a LTL$_f$ formula ϕ, σ satisfies ϕ, denoted by $\sigma \models \phi$, iff $\sigma[0] \models \phi$ where:

- $\sigma[i] \models \texttt{true}$
- $\sigma[i] \models \texttt{final}$ iff $i = n$.
- $\sigma[i] \models \varphi$, where $\varphi \in \mathcal{F}$ iff $\varphi \in s_i$.
- $\sigma[i] \models \neg\phi$ iff $\sigma[i] \not\models \phi$.
- $\sigma[i] \models \phi \wedge \psi$ iff $\sigma[i] \models \phi$ and $\sigma[i] \models \psi$.
- $\sigma[i] \models \bigcirc\phi$ iff $i < n$ and $\sigma_{i+1} \models \phi$
- $\sigma[i] \models \phi \,\mathsf{U}\, \psi$ iff $\exists j : i \leq j \leq n$ such that $\sigma_j \models \psi$ and $\forall k : i \leq k < j : \sigma_k \models \phi$

Additionally, the following standard temporal operators are used:

- Eventually: $\Diamond\phi := \texttt{true}\ \mathsf{U}\ \phi$,
- Always: $\Box\phi := \neg\Diamond\ \neg\phi$,
- Release: $\phi\ \ \mathsf{R}\ \ \psi\ := \neg(\neg\phi\ \mathsf{U}\ \neg\psi)$,
- Weak Until: $\phi\ \ \mathsf{W}\ \ \psi\ := (\phi\ \ \mathsf{U}\ \psi) \vee \Box\phi$.

The size of a LTL$_f$ formula $|\varphi|$ is defined as the number of subformulas.

$$|\varphi| = \begin{cases} 1 & \varphi \in \mathcal{F} \cup \{\texttt{final}, \texttt{true}\} \\ 1 + |\varphi_l| + |\varphi_r| & \varphi = \varphi_l \odot \varphi_r, \odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \mathsf{U}, \mathsf{W}, \mathsf{R}\} \\ 1 + |\varphi'| & \varphi = \odot\varphi', \odot \in \{\neg, \Box, \Diamond\} \end{cases}$$

Approaches that enforce the satisfaction of temporal preferences generally do so by first computing an automaton A_ϕ that accepts the formula ϕ. Here we use an approach that based on non-deterministic finite automata [Baier and McIlraith, 2006b, Edelkamp, 2006].

DEFINITION 11: NON-DETERMINISTIC FINITE AUTOMATA

A **non-deterministic finite automata** (NFA) for a LTL_f formula is a tuple $A_\phi = (Q, \Sigma, \delta, q_0, Q_a)$ where

- Q is a finite set of states
- $\Sigma \subseteq \mathcal{P}(\mathcal{F})$ contains all subsets of facts in ϕ
- $\delta : S \times \mathcal{L}(\mathcal{F}) \mapsto \mathcal{P}(S)$ is the transition relation, where $\mathcal{L}(\mathcal{F})$ is the set of propositional formulae over $\mathcal{F}$
- $q_0 \in Q$ is the initial state
- $Q_a \subseteq Q$ a set of accepting states.

Executing an automaton A_ϕ on a trace $\sigma = s_0 s_1 \cdots s_n \in \Sigma^*$ results in a sequence of automaton states $q_0 q_1 \cdots q_n$, where $(q_i, \phi_i, q_{i+1}) \in \delta$ and $s_i \models \phi_i$. The execution is accepted if $q_n \in Q_a$. There might be multiple executions for a given trace σ. A trace is accepted if there exists an accepted execution.

CHAPTER 4

EXPLANATIONS BASED ON GOAL CONFLICTS

Planning is typically a non-collaborative process. When given a model, a goal, and an optimization objective the planner provides one plan. However, this approach is in many real-life scenarios not applicable. Optimization objectives are often complex and/or implicit in the heads of human users. For example, the year-long experience of space mission planners and their individual preferences [Smith, 2012] cannot easily be reflected. Smith [2012] proposed an *iterative process*, in which the system provides sample plans, allowing the human user to refine their preferences and interactively tailor the plan accordingly.

In such an iterative process, explanations are crucial to support the user. Especially, questions of the form "Why does sample plan π not achieve Q?" need to be answered [Smith, 2012, Krarup et al., 2021]. To answer these questions, insights about the space of possible plans are necessary. We propose addressing these questions through conflicting goals or preferences, such as "Because to achieve Q is necessary to forgo A." or going back to our running example "Because it is not possible to take the crater image when the ice surface soil sample is collected."

Krarup et al. [2021] present an alternative approach to addressing the question "Why does sample plan π not achieve Q?". Their method involves generating a plan π' that satisfies Q and then providing an explanation based on a comparison between π and π'. The disadvantage of this approach is that there might be differences between π and π' that are unrelated to Q. In contrast, our approach replaces the comparison with a *single* alternative satisfying Q with an analysis of common properties of *all* alternative plans. This approach can be seen as a *universal* variant of contrastive plan explanations.

In the following, we introduce a framework for explanations based on conflicts between soft goals in oversubscription planning [Smith, 2004, Domshlak and Mirkis, 2015]. More precisely, we use goal conflicts of the form $Q \rightarrow \neg A$, meaning that all plans satisfying Q cannot satisfy all goals in A. These conflicts are then used to formalize explanations based on the soft goals that are satisfied and not satisfied by the sample plan in the current iteration step. A compact representation of the needed goal conflicts are *minimal unsolvable goal subsets* (MUGS). These are analogous to minimal unsatisfiable core/subsets of constraints in constraint satisfaction and SAT [Liffiton and Sakallah, 2008, Marques-Silva, 2010, Gamba

et al., 2023].

We present two algorithmic approaches to compute all MUGS for a given OSP task. The first approach involves successively exploring the space of soft goal subsets, using either an explicit or a symbolic planner to test the solvability of each individual goal subset. The second approach involves an exhaustive exploration of the state space using a branch-and-bound approach, where all maximal solvable goal subsets are maintained. To prune states where the solvable goal subsets cannot be improved, we leverage approximations based on admissible classical planning heuristics.

To access the feasibility of MUGS computation we provide an empirical analysis on the international planning competition (IPC) benchmarks extended with cost bounds as implemented by Katz et al. [2019] for OSP.

More expressive preferences such as orderings, e. g. "The X-ray image of the crater group should be taken before the soil sample of the ice surface?" enable more nuanced questions and explanations. To support these we build on existing work on temporal preference compilation into singleton goal facts [Edelkamp, 2006, Baier et al., 2009]. Extending a collection of IPC domains with temporal preferences, we analyze how far our approach scales for conflicts between more powerful preferences.

Finally, to evaluate the usefulness of the proposed explanations are in an iterative planning process, we conducted a user study. For this purpose, we first implemented the IPEXCO web tool for Iterative Planning with EXplanations of COnflicts. Then, we conducted an online user study in which we examined users' performance in finding a solvable goal subset that maximizes a given preference metric, with and without access to explanations.

Papers and Contributions The chapter is based on three papers. The framework and the algorithms for MUGS computation are covered by:

Rebecca Eifler, Michael Cashmore, Jörg Hoffmann, Daniele Magazzeni and Marcel Steinmetz
A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning.
Proceedings of the 34th AAAI Conference on Artificial Intelligence (2020)

Rebecca Eifler, Marcel Steinmetz, Alvaro Torralba and Jörg Hoffmann
Plan-Space Explanation via Plan-Property Dependencies: Faster Algorithms & More Powerful Properties.
Proceedings of the 29th International Joint Conference on Artificial Intelligence (2020)

These papers were principally developed by the author in joint work with Marcel Steinmetz, Alvaro Torralba, Jörg Hoffmann, Michael Cashmore and Daniele Magazzeni. The framework is designed based on ideas proposed by Jörg Hoffmann and Daniele Magazzeni for the project "Explaining the Space of Plans" funded by AFOSR. The adaption necessary for symbolic search and the respective implementation was done by Alvaro Torralba. The formalization of nogood and trap learning and their implementations were done by Marcel Steinmetz. The remaining implementation of the search algorithms and the LTL_f and Action-

Set soft goal compilations were done by the author. The evaluation of all approaches are the author's work. Michael Cashmore provided support in the benchmark design.

The iterative planning tool and the user study are covered by the paper:

Rebecca Eifler, Martim Brandao, Amanda Coles, Jeremy Frank and Jörg Hoffmann
Evaluating Plan-Property Dependencies: A Web-Based Platform and User Study.
Proceedings of the 32th International Conference on Automated Planning and Scheduling (2022)

This paper was principally developed by the author in joint work with Martim Brandao, Jeremy Frank, Jörg Hoffmann and Amanda Coles. The implementation of the web tool IPEXCO is the author's work. The user study instances were design by the author. The author conducted the study as well as the quantitative evaluation. Martim Brandao advised the user study setup and provided the analysis of the free text questions.

4.1 EXPLANATIONS

Assume an OSP task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$. In order to facilitate iterative planning, our idea is to explain how the soft goals G^{soft} conflict with each other. The explanations provided in the iterative planning process then take the form of user questions "Why does the plan π you suggest not satisfy my goal $g \in G^{\text{soft}}$?" and are answered with "Because achieving g would necessitate to forego your goal $g' \in G^{\text{soft}}$." Beyond such binary mutexes between goal facts, we address arbitrarily large conflicts, involving any number of goal facts.

Importantly, while we define our framework at the level of goal facts, its use extends to arbitrary goals. Temporal plan preferences p as in PDDL3 [Gerevini et al., 2009], or more generally any *plan property* function p that maps action sequences π in τ to Boolean values, can be addressed, as long as these can be compiled into goal facts. For such extended use, we assume that each plan property p was compiled into a goal fact g_p, modifying and extending τ such that $p(\tau, \pi) = true$ iff $g_p \in I[\![\pi]\!]$[1]. The set G^{soft} of soft goals thus represents the set of plan properties which do not absolutely have to be satisfied (they are not hard goals), but which the user is interested in, and whose trade-offs they wish to explore in the iterative planning process.

We next spell out our conceptual framework. Then we describe the application in an iterative planning process. To facilitate its implementation, we then identify an equivalence to minimal unsolvable goal subsets.

4.1.1 CONCEPTUAL FRAMEWORK

We derive our notion of goal conflicts, and therewith the supported question/answer explanations, via a general concept of plan-space entailment.

[1] Our current implementation supports this process for plan properties p formalized in finite linear temporal logic. We do so based on known techniques described in Section 4.3.

DEFINITION 12: PLAN-SPACE ENTAILMENT

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.
We say that $\pi \in \Pi$ **satisfies** a propositional formula ϕ over G^{soft}, written $\pi \models \phi$, if ϕ evaluates to $true$ under the truth value assignment where $g \in G^{\text{soft}}$ is $true$ iff $g \in I[\![\pi]\!]$.
We denote by $\mathcal{M}_\Pi(\phi) := \{\pi \mid \pi \in \Pi, \pi \models \phi\}$ the subset of plans that satisfy ϕ. We say that ϕ **Π-entails** ψ, written $\Pi \models \phi \to \psi$, if $\mathcal{M}_\Pi(\phi) \subseteq \mathcal{M}_\Pi(\psi)$.

This definition views Π in the role traditionally taken by a knowledge base, identifying a set of "possible worlds" within which the entailment between formulas is considered. Π-entailment is stronger than standard entailment: $\phi \Rightarrow \psi$ implies that $\Pi \models \phi \to \psi$, but not vice versa. Π-entailment captures entailments specific to the space of plans Π, and therewith to the planning task context. This is needed for entailment over goals to be meaningful as goals will rarely entail other goals in the standard logical sense. For an example of plan-space entailment see Example 4.

EXAMPLE 4: PLAN-SPACE ENTAILMENT

In our running example, we consider all goals as soft goals. With an initial battery level of 8 we have $\Pi \models \texttt{image-ice} = \texttt{uploaded} \to \neg(\texttt{soil-sample-rock} = \texttt{uploaded})$. If we achieve $\texttt{image-ice} = \texttt{done}$ then only 5 units are left which is not sufficient to drive to the rock and collect the soil sample. If we set the initial energy level to 9, on the other hand, then the knowledge base changes – Π includes more plans – and that entailment no longer holds.

From a general perspective, Definition 12 specializes concepts from model checking [Clarke et al., 2018] to our setting. The planning task τ takes the role of the model, the plans $\pi \in \Pi$ are the model's execution traces, and $\Pi \models \phi \to \psi$ means that $\tau \models \phi^T \to \psi^T$ where ϕ^T (ψ^T) is a temporal formula expressing that ϕ (ψ) is true at the end of the finite plan trace (this kind of statement is supported by LTL_f [Baier et al., 2009, De Giacomo et al., 2014]). Our simpler special-case notation here captures exactly what is needed in our approach.

Deciding about Π-entailment has the same complexity as deciding plan existence:

PROPOSITION 1: COMPLEXITY OF PLAN-SPACE ENTAILMENT

Deciding whether $\Pi \models \phi \to \psi$ for propositional formulas over soft goals in OSP tasks is **PSPACE**-complete.

Proof:
Both hardness and membership can be shown from the respective properties of deciding FDR plan existence (direct from [Bylander, 1994]).
For hardness, assume an FDR task $\tau = (V, A, c, I, G)$, and consider the OSP task

$\tau' := (V, A, c', I, \emptyset, G, 0)$ where c' assigns all actions cost 0. $\Pi \models \bigwedge_{g \in G} \rightarrow \mathit{false}$, i. e. $\mathcal{M}_\Pi(\bigwedge_{g \in G}) = \emptyset$ holds in τ' iff τ is unsolvable.
For membership, FDR plan existence can be decided via a non-deterministic algorithm that iteratively enumerates all paths through the state space, remembering only a single state s at a time and maintaining a path-length counter l. The algorithm stops when $l = \Pi_{v \in V} |D_v|$ or when $G \subseteq s$. The same algorithm decides whether or not $\Pi \models \phi \rightarrow \psi$, by replacing the test $G \subseteq s$ with $s \models \phi \wedge \neg\psi$.

□

We use plan-space entailment to define goal conflicts, as follows:

DEFINITION 13: GOAL CONFLICT

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.
A **goal conflict (GC)** is an entailment of the form $\Pi \models \bigwedge_{g \in Q} g \rightarrow \neg \bigwedge_{g \in A} g$ where $Q, A \subseteq G^{\text{soft}}$, $Q \neq \emptyset$, and $Q \cap A = \emptyset$. For brevity, we write goal conflicts as $Q \rightarrow_\Pi \neg A$.
We say that goal conflict $Q' \rightarrow_\Pi \neg A'$ **dominates** goal conflict $Q \rightarrow_\Pi \neg A$ if $Q' \subseteq Q$ and $A' \subseteq A$. We say that $Q \rightarrow_\Pi \neg A$ is **dominant** if it is not dominated by any other goal conflict.

We will discuss the first part of this definition, its meaning and implications below. First we address the definition of dominance. This corresponds to the logical strength of the goal conflict, in the following sense:

PROPOSITION 2: DOMINANCE

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, Π its set of plans and $Q, Q', A, A' \subseteq G^{\text{soft}}$ where $Q \cap A = \emptyset$ and $Q' \cap A' = \emptyset$.
If $Q' \rightarrow_\Pi \neg A'$ dominates $Q \rightarrow_\Pi \neg A$, then $Q' \rightarrow_\Pi \neg A'$ is stronger than $Q \rightarrow_\Pi \neg A$:

$$\Pi \vDash (\bigwedge_{g \in Q'} g \rightarrow \neg \bigwedge_{g \in A'} g) \rightarrow (\bigwedge_{g \in Q} g \rightarrow \neg \bigwedge_{g \in A} g)$$

Proof:
By standard logical transformations we can reduce the implication as follows

$$
\begin{aligned}
&(\bigwedge_{g\in Q'} g \rightarrow \neg \bigwedge_{g\in A'} g) \rightarrow (\bigwedge_{g\in Q} g \rightarrow \neg \bigwedge_{g\in A} g)\\
\equiv&(\bigwedge_{g\in Q'} g \wedge \bigwedge_{g\in A'} g) \vee (\neg \bigwedge_{g\in Q} g \vee \neg \bigwedge_{g\in A} g)\\
\equiv& \bigwedge_{g\in Q'\cup A'} g \vee \neg \bigwedge_{g\in Q\cup A} g\\
\equiv& \bigwedge_{g\in Q\cup A} g \rightarrow \bigwedge_{g\in Q'\cup A'} g
\end{aligned}
$$

If $Q' \subseteq Q$ and $A' \subseteq A$, then the last formula is true, and is in particular Π-entailed.

□

Intuitively, dominating conflicts are stronger by the nature of standard entailment. Smaller Q' means weaker left-hand side (smaller conjunction), while smaller A' means stronger right-hand side (smaller disjunction, after moving the negation inside).

Note here the use of *standard entailment*, not taking into account the "knowledge base" Π. Indeed, the opposite direction of Proposition 2 does not hold: $Q' \rightarrow_\Pi \neg A'$ may be stronger than $Q \rightarrow_\Pi \neg A$ even though $Q' \not\subseteq Q$ or $A' \not\subseteq A$. This is because, in plan space, the necessary entailment $\Pi \models \bigwedge_{g\in Q\cup A} g \rightarrow \bigwedge_{g\in Q'\cup A'} g$ may hold anyhow if the truth of goals in $Q \cup A$ Π-entails the truth of other goals $g \in Q' \cup A' \setminus (Q \cup A)$. For example `image-ice` (taking the image of the ice surface) Π-entails `visit-ice` (visiting the ice surface at some point).

Let us now turn to the first part of Definition 13, and therewith its conceptual motivation and nature. Goal conflicts, as defined here, capture exclusion relations between sets of goals. These relations are of interest in OSP where, modulo the precise soft-goal preferences, the task is to achieve as many goals as possible. Indeed, we assume that understanding and navigating goal conflicts is an important aspect of the user's quest for a good plan in iterative planning.

From a conceptual and technical perspective, the following notes are in order.

- The requirement $Q \cap A = \emptyset$ in Definition 13 is natural as it would make no intuitive sense for a goal to be in conflict with (part of) itself.

- The special case $Q = \emptyset$ is excluded from Definition 13 (as well as from Definition 15, discussed below) as this is not a meaningful "conflict" between left-hand side and right-hand side.

- The rim case $A = \emptyset$ is permitted in Definition 13 however (and therewith can be an answer in Definition 15). This makes sense because $\neg \bigwedge_{g\in\emptyset} g$ is equivalent to $false$, so that the entailment $\Pi \models \bigwedge_{g\in Q} g \rightarrow \neg \bigwedge_{g\in A} g$ in this case means that Q is unsolvable in the OSP task τ. This case can naturally appear in iterative planning –

the user's question may be about a set of goals that cannot be achieved conjunctively at all – and is therefore handled as a possible case in our framework.

- Finally, deciding about goal conflicts has the same complexity as deciding about Π-entailment (the arguments in the proof of Proposition 1 remain valid).

The following section describes the concrete iterative planning setup we address, as well as how goal conflicts are leveraged as explanations.

4.1.2 Explanations in Iterative Planning

We instantiate an iterative planning process as described by Smith [2012]. At each iteration step the user determines a set of soft goals the next sample plan must satisfy. The user can then request explanations by asking questions regarding the soft goals not satisfied by the sample plan. Based on the insights gained, the user refines their preferences, which is then reflected in the set of enforced soft goals in the next iteration step. One such iteration step is defined as follows:

Definition 14: Iteration Step

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task with $G^{\text{soft}} \neq \emptyset$ and $G^{\text{enf}} \subseteq G^{\text{soft}}$. An **iteration step** is a pair $\delta = (G^{\text{enf}}, \pi)$, where if task $\tau' = (V, A, c, I, G^{\text{hard}} \cup G^{\text{enf}}, G^{\text{soft}} \setminus G^{\text{enf}}, b)$ is solvable then π is a plan of τ' and $\pi = \epsilon$ the empty sequence ϵ otherwise.

The soft goals G^{enf} are selected by the user, to be then enforced in the sample plan generation, i. e. moving G^{enf} from the soft to the hard goals. Assuming $G^{\text{hard}} \cup G^{\text{enf}}$ is solvable, this results in a sample plan π that satisfies at least $G^{\text{hard}} \cup G^{\text{enf}}$. With $G^{\text{true}}(\pi) = \{g \mid g \in G^{\text{soft}} \wedge g \in I[[\pi]]\}$ we denote the set of soft goals satisfied by π, which are a super set of the enforced goals $G^{\text{enf}} \subseteq G^{\text{true}}(\pi)$ and with $G^{\text{false}}(\pi) = G^{\text{soft}} \setminus G^{\text{true}}(\pi)$ the set of unsatisfied soft goals, which are a subset of the not enforced goals, $G^{\text{false}}(\pi) \subseteq G^{\text{soft}} \setminus G^{\text{enf}}$.

Given a plan π exists, our explanation approach allows users to ask questions about the unsatisfied soft goals $G^{\text{false}}(\pi)$ as follows:

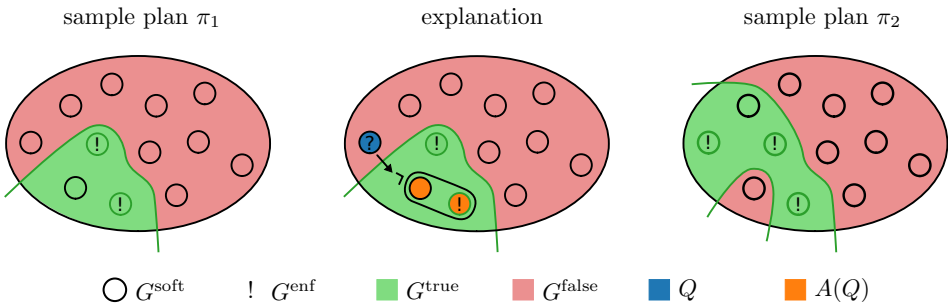
Definition 15: Conflict Explanation

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, Π its set of plans, and $\delta = (G^{\text{enf}}, \pi)$ the current iteration step, where $\pi \neq \epsilon$. Denote by $\min_{\subseteq} \mathcal{S}$ for a set $\mathcal{S}$ of sets the set-inclusion minimal element sets $S \in \mathcal{S}$.
A **user question** is a subset $Q \subseteq G^{\text{false}}(\pi)$ where $Q \neq \emptyset$. The **answer** to such a question is $\mathcal{A}_C(Q) := \min_{\subseteq}\{A' \mid Q' \rightarrow_{\Pi} \neg A'$ is a dominant GC, $Q' \subseteq Q, A' \subseteq G^{\text{true}}(\pi)\}$. We refer to the pair $(Q, \mathcal{A}_C(Q))$ as **conflict explanation**.

The question-answer pairs in a natural-language formulation are "Why does the plan π you suggest not satisfy my goals Q?" – "Because achieving Q would necessitate to forego one goal in each set in $\mathcal{A}_C(Q)$." This is a form of contrastive explanation, based on a

what-if analysis where a set Q of goals requested by the user is hypothetically enforced. An example iteration step with an explanation is given in Example 5.

EXAMPLE 5: ITERATION STEP WITH EXPLANATION



Example plan π_1 satisfies three soft goals, two of which are enforced (left). Then the user asks a question $Q = \{$?$\}$ by selecting a subset of G^{false} (center). The answer $A(Q) = \{\{$●, !$\}\}$ consists of two soft goals in G^{true} and means: "If you want to satisfy ? then you have to forego one of $\{$●, !$\}$". For the next iteration step the user decides that forging ● is acceptable (as it was not enforced in the first place) and enforces ?. Then they get the next sample plan π_2 (right).

Regarding the concept of user questions and the corresponding answers, note the following:

- The rim case $Q = \emptyset$ is excluded from Definition 15 as this is not a meaningful user question.
- The definition of answers furthermore mirrors the definition of dominance in Definition 13, in the sense that the condition $Q \supseteq Q'$ uses the same simplified notion of "stronger than": we use standard entailment (which boils down to set inclusion)
- Answers only include soft goals that are currently true $(A' \subseteq G^{\text{true}}(\pi))$. One could argue that telling the user what needs to stay false to include Q, could be beneficial too. However, this information does not address the user's question, of why Q is not satisfied. We assume that the user includes all soft goals they want to be considered in the question or ask multiple questions to cover all goals of interest.
- We return only the minimal answers A' because these are the strongest replies to the user question. For example, if $\{q_1\} \rightarrow_\Pi \neg\{p, r\}$ and $\{q_1, q_2\} \rightarrow_\Pi \neg\{p\}$ are dominant goal conflicts, then including both $\{p, r\}$ and $\{p\}$ in the answer to question $Q = \{q_1, q_2\}$ would be misleading.
- When answering questions we do not specify which part $Q' \subseteq Q$ causes the conflict for each A'. This additional information again does not address the user's question. If they are interested in the conflicts of a particular subset of Q, we assume them to phrase their questions accordingly.

Depending on whether the question Q is part of a goal conflict, the answer may be interpreted in different ways. Figure 5 depicts the three possible cases.

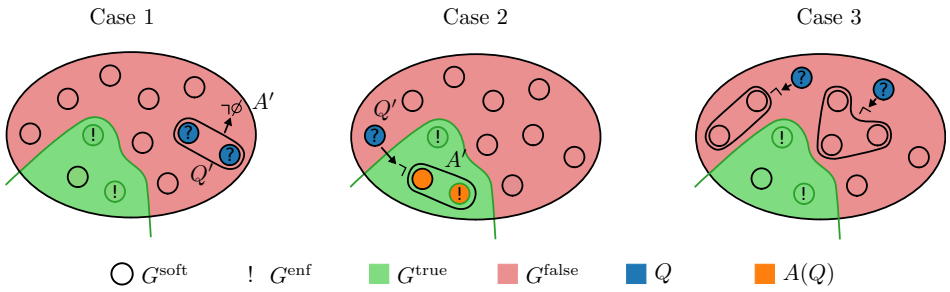


Figure 5: Different possible questions Q and answers $\mathcal{A}_C(Q)$ depending on their relation to a dominant goal conflict.

Case 1:
There exists a dominant GC $Q' \rightarrow_\Pi \neg A'$ such that $Q' \subseteq Q$ and $A' = \emptyset$. Then the answer is $\mathcal{A}_C(Q) = \{\emptyset\}$, meaning: "Achieving Q itself is not possible." This case happens if the user is not aware that achieving the soft goals in their questions Q is not possible.

Case 2:
There exists a dominant GC $Q' \rightarrow_\Pi \neg A'$ such that $Q' \subseteq Q, A' \subseteq G^{\text{true}}(\pi)$ and $A' \neq \emptyset$. Then the answer is $\mathcal{A}_C(Q) = \{A'\}$ meaning: "Because achieving Q would necessitate to forego one of the soft goals in A'." This is the "normal" case where the user wants to include additional soft goals, but has to forego satisfied soft goals to do so.

Case 3:
There exists **no** dominant GC $Q' \rightarrow_\Pi \neg A'$ such that $Q' \subseteq Q, A' \subseteq G^{\text{true}}(\pi)$. Then the answer is $\mathcal{A}_C(Q) = \{\}$, meaning "Achieving Q is possible." It can happen that $Q \cup G^{\text{true}}(\pi)$ can be satisfied and thus Q could be enforced without foregoing any currently true soft goal.

If Q can be satisfied without foregoing any goal facts (Case 3), the question of why the plan does not satisfy Q in the first place might arise. The only objective of a plan is to satisfy $G^{\text{hard}} \cup G^{\text{enf}}$. Any additional soft goals that are satisfied are either a coincidence or implied by one of the enforced soft goals. Including more soft goals in the plan than the user specified would imply that satisfying more soft goals is always preferable, though this is not necessarily the case. If the user wants to ensure satisfaction, they have the option to enforce them. The automatic inclusion of soft goals could potentially result in unnecessary overhead, as the answer size is increased by soft goals that are not currently of interest to the user.

Unsolvable Task If $G^{\text{hard}} \cup G^{\text{enf}}$ is not solvable then there exists no plan for iteration step $\delta = (G^{\text{enf}}, \epsilon)$. Goal conflicts can also provide an explanation in this case, as they represent the information as to which subsets of G^{enf} cause the unsolvability.

DEFINITION 16: EXPLANATIONS OF UNSOLVABILITY

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and $\delta = (G^{\text{enf}}, \pi)$ the current iteration step, where $\pi = \epsilon$.
$\mathcal{A}\!\!\not\!\!\ell = \{G \mid G \rightarrow_\Pi \neg\emptyset$ is a dominant GC, $G \subseteq G^{\text{enf}}\}$.

The question "Why is there no plan for G_i^{enf}?" can be answered by "Because the soft goal subsets $\mathcal{A}\!\!\not\!\!\ell$ cannot be satisfied". We do provide all goal conflicts contained in G_i^{enf}, since they must all be addressed in order to obtain a solvable selection of enforced hard goals G_{i+1}^{enf} for the next iteration step.

4.1.3 EQUIVALENCE TO MUGS

We now show that our notion of explanations can be equivalently formulated via unsolvable goal subsets, facilitating its implementation. Dominant goal conflicts can alternatively be characterized as minimal unsolvable goal subsets.

DEFINITION 17: MUGS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task.
We say that $G \subseteq G^{\text{soft}}$ is a **minimal unsolvable goal subset (MUGS)** if planning task $\tau' = (V, A, c, I, G^{\text{hard}} \cup G, \emptyset, b)$ is unsolvable but for all $G' \subset G$ $\tau'' = (V, A, c, I, G^{\text{hard}} \cup G', \emptyset, b)$ is solvable.
By $\mathcal{G}^{\text{MUGS}}(\tau) := \{G \mid G \subseteq G^{\text{soft}}, G$ is a MUGS$\}$ we denote the set of all MUGS in τ.

PROPOSITION 3: DOMINANT GOAL CONFLICT $\Leftrightarrow$ MUGS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.
Then, for any $Q, A \subseteq G^{\text{soft}}$ where $Q \neq \emptyset$ and $Q \cap A = \emptyset$, $Q \rightarrow_\Pi \neg A$ is a dominant GC iff $Q \cup A \in \mathcal{G}^{\text{MUGS}}(\tau)$.

Proof:
A Π-entailment $\Pi \models \bigwedge_{g \in Q} g \rightarrow \neg \bigwedge_{g \in A} g$ holds iff $Q \cup A$ is unsolvable, i. e., if there is no plan for $(V, A, c, I, G^{\text{hard}} \cup Q \cup A, \emptyset, b)$. Dominant entailments result from set-inclusion minimal Q and A, corresponding to the set-inclusion minimality of MUGS.

□

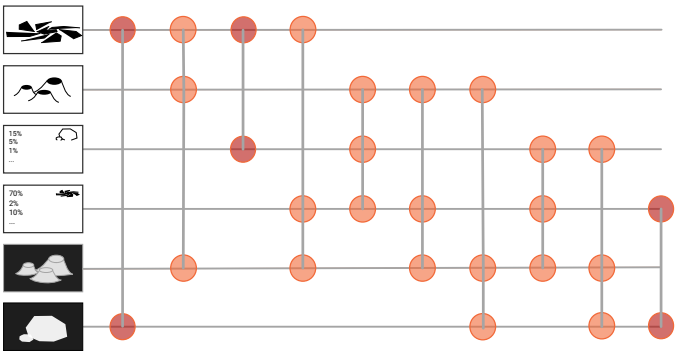
The number of MUGS can be exponentially larger than the number of soft goals. All MUGS in our running example are depicted in Example 6.

EXAMPLE 6: MUGS

In the following list we represent the soft goal $x = \texttt{uploaded}$ simply with x. Our running example has 10 MUGS with 2 or 3 elements.

- $\{\texttt{image-crater3}, \texttt{x-ray-image-rock}, \texttt{x-ray-image-crater3}\}$
- $\{\texttt{image-crater3}, \texttt{x-ray-image-crater3}, \texttt{soil-sample-ice}\}$
- $\{\texttt{x-ray-image-rock}, \texttt{soil-sample-ice}\}$
- $\{\texttt{image-ice}, \texttt{x-ray-image-rock}\}$
- $\{\texttt{image-crater3}, \texttt{soil-sample-rock}, \texttt{soil-sample-ice}\}$
- $\{\texttt{image-ice}, \texttt{image-crater3}, \texttt{x-ray-image-crater3}\}$
- $\{\texttt{x-ray-image-crater3}, \texttt{soil-sample-rock}, \texttt{soil-sample-ice}\}$
- $\{\texttt{image-ice}, \texttt{x-ray-image-crater3}, \texttt{soil-sample-ice}\}$
- $\{\texttt{x-ray-image-rock}, \texttt{x-ray-image-crater3}, \texttt{soil-sample-rock}\}$
- $\{\texttt{image-ice}, \texttt{soil-sample-rock}\}$

In the following visualization, each column corresponds to one MUGS. It shows that `x-ray-image-crater3` is involved in the most conflicts, while `image-ice` is part of two conflicts of size 2 with the data points of the rock. This is not surprising, since the rock is on the other side of the map.



Given this, our desired explanations can be provided based on MUGS.

THEOREM 1: EXPLANATIONS FROM MUGS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and $\delta = (G^{\text{enf}}, \pi)$ the current iteration step, where $\pi \neq \epsilon$. Let $\emptyset \neq Q \subseteq G^{\text{false}}(\pi)$ be a user question.
Then $\mathcal{A}_C(Q) = \min_{\subseteq}\{G \setminus Q \mid G \in \mathcal{G}^{\text{MUGS}}(\tau), G \subseteq Q \cup G^{\text{true}}(\pi)\}$.

Proof:
Recall that $\mathcal{A}_C(Q) = \min_{\subseteq}\{A' \mid Q' \rightarrow_\Pi \neg A'$ is a dominant GC, $Q' \subseteq Q, A' \subseteq G^{\text{true}}(\pi)\}$. In what follows, we prove that $\{A' \mid Q' \rightarrow_\Pi \neg A'$ is a dominant GC, $Q' \subseteq Q$, $A' \subseteq G^{\text{true}}(\pi)\} = \{G \setminus Q \mid G \in \mathcal{G}^{\text{MUGS}}(\tau), G \subseteq Q \cup G^{\text{true}}(\pi)\}$, dropping the reduction to set-inclusion minimal members from both sets. The claim then follows from:
$\subseteq$: Let $Q' \rightarrow_\Pi \neg A'$ be a dominant goal conflict where $Q' \subseteq Q$ and $A' \subseteq G^{\text{true}}(\pi)$. By Proposition 3 $G := Q' \cup A'$ is a MUGS. With $Q' \subseteq Q$ and $A' \subseteq G^{\text{true}}(\pi)$, we have that $G = Q' \cup A' \subseteq Q \cup G^{\text{true}}(\pi)$. It remains to show that $G \setminus Q = A'$, i. e., that $(Q' \cup A') \setminus Q = A'$; this follows directly from $Q' \subseteq Q$ and $Q \cap A' = \emptyset$ which holds because of $Q \subseteq G^{\text{false}}(\pi)$ and $A' \subseteq G^{\text{true}}(\pi)$.
$\supseteq$: Let G be a MUGS where $G \cap Q \neq \emptyset$. Define $Q' := G \cap Q$ and $A' := G \setminus Q'$. By Proposition 3 and because $Q' = G \cap Q \neq \emptyset$, $Q' \rightarrow_\Pi \neg A'$ is a dominant goal conflict. We have $Q' \subseteq Q$ by construction. With $G \subseteq Q \cup G^{\text{true}}(\pi)$ and $Q \cap G^{\text{true}}(\pi) = \emptyset$ we have $A' = G \setminus Q' = G \setminus (G \cap Q) = G \setminus Q \subseteq (Q \cup G^{\text{true}}(\pi)) \setminus Q = G^{\text{true}}(\pi)$. It remains to show that $A' = G \setminus Q$, i. e., that $G \setminus Q' = G \setminus Q$; this follows directly from $Q' = G \cap Q$.
□

The three cases of answers as introduced for goal conflicts in Figure 5 can be applied to MUGS as shown in Figure 6. For Case 1 there exists a MUGS G such that $G \subseteq Q$; for Case 2 there exists a MUGS G such that $G \setminus Q \subseteq G^{\text{true}}(\pi)$; and for Case 3 for all MUGS $G : G \setminus (Q \cup G^{\text{true}}) \neq \emptyset$ holds. MUGS answer the question of why a set of enforced soft goals G^{enf} leads to an unsolvable task with $\mathcal{A}\not{\iota} = \{G \mid G \in \mathcal{G}^{\text{MUGS}}(\tau), G \subseteq G^{\text{enf}}\}$.

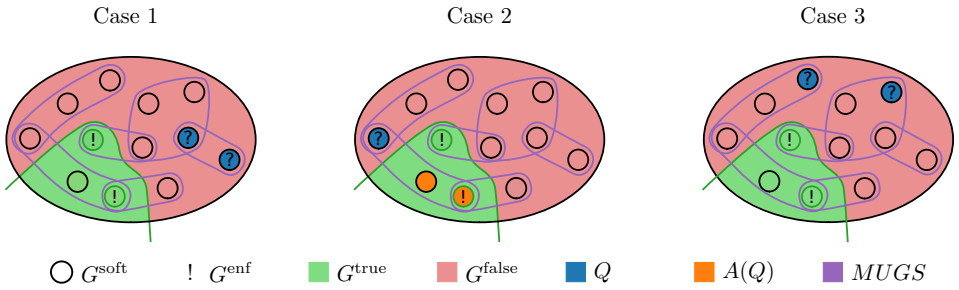


Figure 6: Different possible questions Q and answers $\mathcal{A}_C(Q)$ depending on their relation to MUGS.

This alternative representation of our explanations is useful because, a single MUGS G represents $2^{|G|} - 1$ dominant goal conflicts: every way of selecting a non-empty Q from G and thus splitting G into the left-hand/right-hand sides Q and A of the goal conflict. Hence, MUGS are our representation of choice for computing dominant goal conflicts. Specifically, our approach is to compute all MUGS i. e., to compute the set $\mathcal{G}^{\text{MUGS}}(\tau)$.

DEFINITION 18: ALLMUGS

Let τ be an OSP task. By **AllMUGS** we denote the algorithmic problem of computing $\mathcal{G}^{\text{MUGS}}(\tau)$.

Through solving AllMUGS, with Theorem 1 we pre-compute the answers to all possible user questions as per Definition 15. This can be done offline, prior to the iterative planning process, which makes sense given that responses to user questions should ideally be instantaneous. Deciding whether a goal set is a MUGS is **PSPACE**-complete given the above (Propositions 1 and 3). We will see later that many of our algorithms solving AllMUGS solve cardinality-optimal oversubscription planning (every soft goal has the same utility) as a side effect.

That said, one can also use Definition 18 in an online setting, computing only the answer to an individual user question Q for a given plan π. This can be done by a simple task modification.

THEOREM 2: INDIVIDUAL ANSWER BY TASK TRANSFORMATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and $\delta = (G^{\text{enf}}, \pi)$ the current iteration step, where $\pi \neq \epsilon$. Let $Q \subseteq G^{\text{false}}(\pi)$ with $Q \neq \emptyset$ be a user question. With $\tau_{Q!} := (V, A, c, I, G^{\text{hard}} \cup Q, G^{\text{soft}} \setminus Q, b)$ the conflict explanation is given by $\mathcal{A}_C(Q) = \{A' \mid A' \in \mathcal{G}^{\text{MUGS}}(\tau_{Q!}), A' \subseteq G^{\text{true}}(\pi)\}$.

Proof:
Leveraging the equivalence identified by Theorem 1, what we need to show is that $\min_{\subseteq}\{G \setminus Q \mid G \in \mathcal{G}^{\text{MUGS}}(\tau), G \subseteq Q \cup G^{\text{true}}(\pi)\} = \{A' \mid A' \in \mathcal{G}^{\text{MUGS}}(\tau_{Q!}), A' \subseteq G^{\text{true}}(\pi)\}$.

$\subseteq$: Let $A' := G \setminus Q$.

- $A' \subseteq G^{\text{true}}(\pi)$ follows from $G \subseteq Q \cup G^{\text{true}}(\pi)$
- $A' \in \mathcal{G}^{\text{MUGS}}(\tau_{Q!})$:
  - A' is unsolvable in $\tau_{Q!}$ follows from $G \in \mathcal{G}^{\text{MUGS}}(\tau)$. If $G \in \mathcal{G}^{\text{MUGS}}(\tau)$ then $G^{\text{hard}} \cup G$ is unsolvable. If A' is unsolvable in $\tau_{Q!}$ then $G^{\text{hard}} \cup Q \cup A' = G^{\text{hard}} \cup Q \cup (G \setminus Q) = G^{\text{hard}} \cup Q \cup G$, which is unsolvable because $G^{\text{hard}} \cup G$ is unsolvable.
  - That A' is minimal, we show with proof by contradiction. Assume A' is not minimal, that means $\exists g \in A' : A' \setminus g$ is unsolvable in $\tau_{Q!}$. This means $(A' \setminus g) \cup Q$ is unsolvable in τ. Thus, there exists $C \in \mathcal{G}^{\text{MUGS}}(\tau)$, such that $C \subseteq (A' \setminus g) \cup Q$. Because of $A \subseteq G^{\text{true}}(\pi)$ and $Q \subseteq G^{\text{false}}(\pi)$ we have $A' \cap Q = \emptyset$ and thus $C \setminus Q \subseteq (A' \setminus g \cup Q) \setminus Q = A' \setminus g \subsetneq A'$. However, $C \setminus Q \subsetneq A'$ implies that if $C \setminus Q$ is in T then A' cannot be in $\min_{\subseteq} T$ and thus A' has to be minimal.

$\supseteq$: Because of $A' \in \mathcal{G}^{\text{MUGS}}(\tau_{Q!})$, $Q \cup A'$ is unsolvable in τ. Let $G \subseteq A' \cup Q$ and $G \in \mathcal{G}^{\text{MUGS}}(\tau)$.

- $G \subseteq Q \cup G^{\text{true}}(\pi)$ holds, because of $A' \subseteq G^{\text{true}}(\pi)$.

- $A' = G \setminus Q$:
  - $A' \supseteq G \setminus Q$ follows from $A' \cup Q \supseteq G$ and $A' \cap Q = \emptyset$.
  - We show with proof by contradiction $A' \subseteq G \setminus Q$. Assume $A' \nsubseteq G \setminus Q$. Because of $G \in \mathcal{G}^{\text{MUGS}}(\tau)$, $G \setminus Q$ is unsolvable in $\tau_{Q!}$, but then $G \setminus Q \subsetneq A'$ is unsolvable in $\tau_{Q!}$. This is not possible, because A' is minimal.
- It remains to show, that set-inclusion minimality does not remove A'. We use a proof by contradiction, assuming that A' is removed. If this is the case, then there exists $A'' \subsetneq A'$ in $\{G \setminus Q \mid G \in \mathcal{G}^{\text{MUGS}}(\tau), G \subseteq Q \cup G^{\text{true}}(\pi)\}$. Because $A' \setminus A'' \nsubseteq Q$ there exists a $G' \subsetneq G$ such that $G' \in \mathcal{G}^{\text{MUGS}}(\tau)$, but this is not possible because G is minimal.

□

An example for an online computation based on Theorem 2 is given in Example 7.

EXAMPLE 7: INDIVIDUAL ANSWER BY TASK TRANSFORMATION

Let's consider our running example again where all data points are soft goals. Assume the current plan π_i does include `image-ice` but none of the rock data points. Thus, the user asks "Why does π_i not include any rock data points?". This gives us the question $Q = \{\texttt{soil-sample-rock}, \texttt{x-ray-image-rock}\}$. To compute an answer online, we compute $\mathcal{G}^{\text{MUGS}}(\tau_{Q!})$ where $G^{\text{hard}} = Q$ is the question and $G^{\text{soft}} = DP \setminus Q$ includes all other data point.
The resulting MUGS:

- $\{\texttt{soil-sample-ice}\}$
- $\{\texttt{x-ray-image-crater3}\}$
- $\{\texttt{image-ice}\}$

give us the answer $\mathcal{A}_C(Q) = \{\texttt{image-ice}\}$.
This is the same answer we get based on all MUGS in $\mathcal{G}^{\text{MUGS}}(\tau)$ that contain a rock data point:

- $\{\texttt{image-crater3}, \texttt{x-ray-image-rock}, \texttt{x-ray-image-crater3}\}$
- $\{\texttt{x-ray-image-rock}, \texttt{soil-sample-ice}\}$
- $\{\texttt{image-ice}, \texttt{x-ray-image-rock}\}$
- $\{\texttt{image-crater3}, \texttt{soil-sample-rock}, \texttt{soil-sample-ice}\}$
- $\{\texttt{x-ray-image-crater3}, \texttt{soil-sample-rock}, \texttt{soil-sample-ice}\}$
- $\{\texttt{x-ray-image-rock}, \texttt{x-ray-image-crater3}, \texttt{soil-sample-rock}\}$
- $\{\texttt{image-ice}, \texttt{soil-sample-rock}\}$

By first removing all rock data points

- {image-crater3, x-ray-image-crater3}
- {soil-sample-ice}
- {image-ice}
- {image-crater3, soil-sample-ice}
- {x-ray-image-crater3, soil-sample-ice}
- {x-ray-image-crater3}
- {image-ice}

and then restoring subset minimality

- {soil-sample-ice}
- {x-ray-image-crater3}
- {image-ice}

4.2 ALLMUGS ALGORITHMS

Next we introduce two algorithmic approaches to compute AllMUGS. The first approach is based on a systematic exploration of the goal subset space, while the second approach uses one exhaustive state-space exploration. Both algorithms use the notion of maximal solvable goal subsets (MSGS).

DEFINITION 19: MSGS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task.
We say that $G \subseteq G^{\text{soft}}$ is a **maximal solvable goal subset (MSGS)** if planning task $\tau' = (V, A, c, I, G^{\text{hard}} \cup G, \emptyset, b)$ is solvable and for all $G' \supset G$ task $\tau'' = (V, A, c, I, G^{\text{hard}} \cup G', \emptyset, b)$ is unsolvable.
By $\mathcal{G}^{\text{MSGS}}(\tau) := \{G \mid G \subseteq G^{\text{soft}}, G \text{ is a MSGS}\}$ we denote the set of all MSGS in τ.

MSGS hence are the dual of MUGS, just like minimal unsatisfiable subsets and maximal satisfiable subsets in CP [Bailey and Stuckey, 2005, Liffiton and Sakallah, 2008].

PROPOSITION 4: MUGS-MSGS RELATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task. The relation between $\mathcal{G}^{\text{MUGS}}(\tau)$ and $\mathcal{G}^{\text{MSGS}}(\tau)$ is given by $\mathcal{G}^{\text{MUGS}}(\tau) = \mathit{HIT}(\overline{\mathcal{G}^{\text{MSGS}}(\tau)})$ and $\mathcal{G}^{\text{MSGS}}(\tau) = \overline{\mathit{HIT}(\mathcal{G}^{\text{MUGS}}(\tau))}$.

Where $\overline{\mathcal{S}} = \{G^{\text{soft}} \setminus S \mid S \in \mathcal{S}\}$ is the complement set with respect to G^{soft} and $\mathit{HIT}(\mathcal{S}) = \min_{\subseteq}\{S \mid \forall S' \in \mathcal{S} : S \cap S' \neq \emptyset\}$ are the *minimal hitting sets* that contain one element from each set in $\mathcal{S}$.

EXAMPLE 8: MUGS VS MSGS

Given $G^{\text{soft}} = \{A, B, C, D\}$ and $\mathcal{G}^{\text{MSGS}}(\tau) = \{\{A, D\}, \{B, C\}, \{B, D\}, \{C, D\}\}$. Then $\overline{\mathcal{G}^{\text{MSGS}}(\tau)} = \{\{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}\}$ and $\mathit{HIT}(\overline{\mathcal{G}^{\text{MSGS}}(\tau)}) = \mathcal{G}^{\text{MUGS}}(\tau) = \{\{A, B\}, \{A, C\}, \{B, C, D\}\}$.

	$\mathcal{G}^{\text{MSGS}}(\tau)$				$\overline{\mathcal{G}^{\text{MSGS}}(\tau)}$				$\mathit{HIT}(\overline{\mathcal{G}^{\text{MSGS}}(\tau)})$		
A				×	×	×	×		×	×	
B		×	×		×			×	×		×
C	×		×			×		×		×	×
D	×	×		×			×				×

	$\mathcal{G}^{\text{MUGS}}(\tau)$			$\mathit{HIT}(\mathcal{G}^{\text{MUGS}}(\tau))$				$\overline{\mathit{HIT}(\mathcal{G}^{\text{MUGS}}(\tau))}$		
A	×	×		×	×	×				×
B	×		×	×				×	×	
C		×	×		×		×		×	
D			×			×	×	×		×

Some of the algorithms introduced below use MSGS as an intermediate step to compute MUGS. We use Proposition 4 to translate MSGS into MUGS. Minimal hitting sets is a known NP-complete decision problem [Karp, 1972]. We use the algorithm for hitting set computation by Berge [Bailey and Stuckey, 2005, Berge, 1989], with the following recursive definition for goal subsets $\mathcal{G} = \{G_0, \cdots, G_n\}$:

$$\mathit{HIT}(\mathcal{G}) = \begin{cases} \{\{g\} \mid g \in G_0\} & \text{if } \mathcal{G} = \{G_0\} \\ \min_{\subseteq}\{H \cup \{g\} \mid g \in G \in \mathcal{G}, H \in \mathit{HIT}(\mathcal{G} \setminus G)\} & \text{otherwise} \end{cases}$$

The equation can be solved in time exponential in $|\mathcal{G}|$.

4.2.1 GOAL-LATTICE SEARCH

For a given OSP task τ with soft goals G^{soft} to determine whether $G \subseteq G^{\text{soft}}$ is a MUGS, you have to check if G is unsolvable and for each proper subset $G' \subsetneq G$ whether G' is solvable. To check these properties for all goal subsets, we systematically traverse the goal lattice composed of all goal subsets.

DEFINITION 20: GOAL LATTICE

Given a set of goals G, the **goal lattice** is defined as $\mathcal{L}(G) = (\mathcal{P}(G), \subseteq)$.

The goal lattice defines a graph over the subsets of soft goals G^{soft}, with edges between $G, G' \subseteq G^{\text{soft}}$ if $G' = G \cup g$ for some $g \in G^{\text{soft}}$. An example of a goal lattice is shown in Figure 7. The (un-)solvability of goal subsets is transitive, i. e. for all $G' \subsetneq G$ if G is solvable then G' is solvable and for all $G' \supsetneq G$ if G is unsolvable then G' is unsolvable. Thus, the

solvable and unsolvable goal subsets partition the goal lattice graph into two parts, with MSGS and MUGS forming the border of goal subsets with only unsolvable super sets and only solvable subsets respectively.

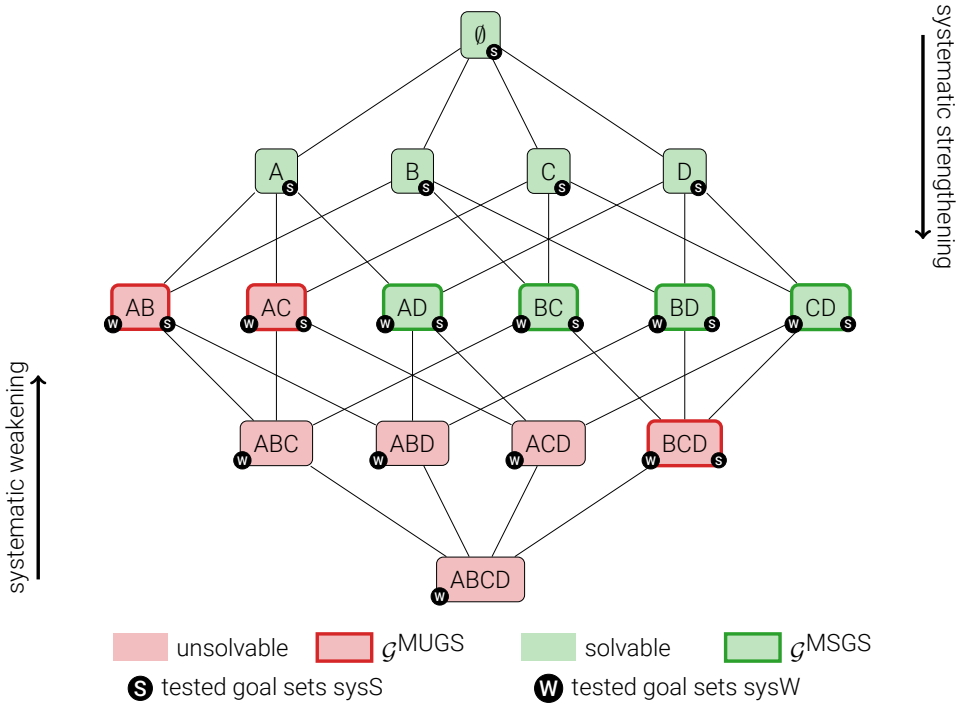


Figure 7: Goal lattice for $G^{\text{soft}} = \{A, B, C, D\}$; AB is the abbreviation for the set $\{A, B\}$. $\mathcal{G}^{\text{MSGS}}(\tau) = \{\{A, D\}, \{B, C\}, \{B, D\}, \{C, D\}\}$ and $\mathcal{G}^{\text{MUGS}}(\tau) = \{\{A, B\}, \{A, C\}, \{B, C, D\}\}$ form the border between the solvable and unsolvable parts.

To find this border, we traverse the goal lattice systematically. Each subset G is tested for solvability. Depending on the result, we use the transitivity property of (un-)solvability to cut off the search. Keeping track of the frontier of solvable and unsolvable goal subsets allows to extract $\mathcal{G}^{\text{MUGS}}(\tau)$ in post-processing. The pseudo-code for goal lattice search (GLS) is given in Algorithm 2.

The order in which goal subsets are tested has a major impact on the number of tests required to determine (un-)solvability for all goal subsets. We implement two complementary expansion orders:

SysW: starting with G^{soft} and systematically weakening the goal subsets by removing goal facts;

SysS: starting with $\emptyset$ and systematically strengthening the goal subsets by adding goal facts.

For both the open list $\mathcal{O}$ is a FIFO queue, leading to a breath first exploration of the goal lattice. The pseudo code for the corresponding expansion functions is given in Algorithm 3. The subsets that need to be tested for SysW and SysS assuming a lexicographical ordering of the children are marked in Figure 7. In SysW, a goal subset is further weakened only

Algorithm 2 Goal-Lattice Search (GLS).

Input: OSP task τ
function GOALLATTICESEARCH(τ, INITSET, EXPAND)
  Solvable $\leftarrow \{\}$
  Unsolvable $\leftarrow \{\}$
  $\mathcal{O} \leftarrow$ INITOPEN(G^{soft}) ▷ *FIFO queue*
  while $|\mathcal{O}| > 0$ **do**
    $G \leftarrow$ POP($\mathcal{O}$)
    isSolvable $\leftarrow$ TESTSOLVABLE($\tau' = (V, A, c, I, G^{\text{hard}} \cup G, \emptyset, b)$)
    if isSolvable **then**
      Solvable $\leftarrow$ *Solvable* $\cup \{G\}$
    else
      Unsolvable $\leftarrow$ *Unsolvable* $\cup \{G\}$
    $\mathcal{O} \leftarrow$ UPDATEOPEN(G, isSolvable, $\mathcal{O}$, *Solvable*, *Unsolvable*)
  return $\{G \in$ *Unsolvable* $\mid \nexists G' \in$ *Unsolvable* : $G' \subset G\}$

if it is unsolvable, since if G is solvable all subsets are solvable too. A goal subset G' is added only if there is no solvable super set that determines that G' is also solvable. The open list is updated to remove all subsets that are now known to be solvable. For example in Figure7, $\{A\}$ is added as a child of $\{A, B\}$, but since $\{A, D\}$ proves to be solvable, $\{A\}$ is also solvable and does not require separate testing. The same rules apply to SysS. Here a subset is only further strengthened if it is solvable and a subset is only added if there is no unsolvable subset.

Algorithm 3 Expansion functions for systematic weakening and strengthening.

function INITOPENSYSW(G)
  return $\{G\}$
function UPDATEOPENSYSW(G, isSolvable, $\mathcal{O}$, *Solvable*, *Unsolvable*)
  if isSolvable **then** ▷ *Prune via solvability transitivity*
    return $\mathcal{O} \setminus \{G' \in \mathcal{O} \mid G' \subset G\}$
  else ▷ *Expand: remove single goal facts*
    $\mathcal{G} \leftarrow \{G \setminus \{g\} \mid g \in G, \forall G' \in$ *Solvable* : $G \setminus \{g\} \not\subseteq G'\}$
    return $\mathcal{O} \cup \mathcal{G}$
function INITOPENSYSS(G)
  return $\{\emptyset\}$
function UPDATEOPENSYSS(G, isSolvable, $\mathcal{O}$, *Solvable*, *Unsolvable*)
  if $\neg$isSolvable **then** ▷ *Prune via unsolvability transitivity*
    return $\mathcal{O} \setminus \{G' \in \mathcal{O} \mid G' \supset G\}$
  else ▷ *Expand: add single goal facts*
    $\mathcal{G} \leftarrow \{G \cup \{g\} \mid g \in G^{\text{soft}} \setminus G, \forall G' \in$ *Unsolvable* : $G \setminus \{g\} \not\supseteq G'\}$
    return $\mathcal{O} \cup \mathcal{G}$

PROPOSITION 5: CORRECTNESS GOAL LATTICE SEARCH

Goal Lattice Search (Algorithm 2) with SysW or SysS computes $\mathcal{G}^{\text{MUGS}}(\tau)$.

Proof:

SysW: On termination, *Unsolvable* contains all unsolvable goal subsets, because if a goal subset is unsolvable, all its children are tested unless solvability can be derived from the subset relation. This means *Solvable*$\cup$*Unsolvable* is only missing solvable subsets. Thus, the return statement $\{G \in \textit{Unsolvable} \mid \nexists G' \in \textit{Unsolvable} : G' \subset G\}$ is equivalent to the MUGS property.

SysS: On termination, (1) *Solvable* contains all solvable goal subsets and (2) $\mathcal{G}^{\text{MUGS}}(\tau) \subseteq$ *Unsolvable*. The former is true, because if a goal subset is solvable, all its children are tested unless unsolvability can be derived from the subset relation. (2) is true, because according to the definition all proper subsets of a MUGS are solvable. Thus, all subsets are considered during the search and each MUGS is pushed to the open queue. Since all subsets are solvable a MUGS can never be pruned from the open queue, thus it will eventually be part of *Unsolvable*. Thus, all MUGS are computed.

□

Worst case the number of subsets checked for solvability is exponential in $|G^{\text{soft}}|$. For SysW all unsolvable goal subsets and $\mathcal{G}^{\text{MSGS}}(\tau)$ and for SysS all solvable goal subsets and $\mathcal{G}^{\text{MUGS}}(\tau)$ are tested (follows from Proof of Proposition 5). Intuitively SysW is better suited if $\mathcal{G}^{\text{MSGS}}(\tau)$ are large and thus encountered early, while SysS is better suited if $\mathcal{G}^{\text{MUGS}}(\tau)$ are large. However, often finding a plan is easier than proving unsolvability, which could give SysS an advantage. A corresponding empirical analysis for the IPC domain is given in Section 4.3.1.

Similar approaches are used in constraint satisfaction to compute minimal unsatisfiable constraint sets. In this context, other exploration functions targeting an anytime approach have been proposed [Liffiton et al., 2016]. Our objective is to compute all MUGS. Since planning is PSPACE-hard, we instead focus on optimizing the individual solvability checks. For further references see Section 4.6.

In GLS, each subset must be tested for solvability. To do so, any off-the-shelf classical planner that can handle cost bounds can be used. However, an explicit state space search has the disadvantage of generating similar search spaces multiple times, which is therefore very inefficient. The only difference between the tasks is the goals, so they all share the same state space. We exploit this in two different ways. First, we use symbolic search for a more efficient solvability check in GLS. Then, we introduce a branch-and-bound approach to compute allMUGS in one exhaustive explicit state space exploration.

SOLVABILITY CHECK WITH SYMBOLIC SEARCH

Contribution: Álvaro Torralba provided the following description and the implementation of the BDD representation of a cost bounded reachable search space.

For a more efficient solvability check, we propose using symbolic search, a search paradigm that uses Binary Decision Diagrams (BDDs) [Bryant, 1986] to represent sets of states, often requiring exponentially less memory than their explicit enumeration. Thus, BDDs are a suitable and compact representation for the set $\mathcal{S}^r$. After computing a BDD β_τ that represents $\mathcal{S}^r$, each test within the goal-lattice search (line 8 in Algorithm 2) can be efficiently performed as detailed below.

Furthermore, symbolic search has been shown to be a powerful tool for state space exhaustion, both in model-checking [McMillan and McMillan, 1993, Burch et al., 1994] and planning [Edelkamp and Kissmann, 2009, Torralba et al., 2017]. In symbolic search, operations on BDDs are used to manipulate sets of states. For example, the disjunction of two BDDs corresponds to computing the BDD that represents the union of the corresponding sets of states, whereas the conjunction represents the intersection. Assembling this machinery for our purposes, the main difference to previous work is that we need to adhere to the cost bound. To this end, given an OSP task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$, we run a symbolic forward uniform-cost search.

Symbolic forward uniform cost-search starts with a BDD, β_0, representing the initial state, I. Then, it iteratively generates the set of states reachable with a cost g for increasing values of $g = 0, \dots, b$. To that end, successor-state generation can be directly implemented on the BDD representation, via BDD operations whose runtime depends only on the BDD size, not on the number of states represented. For a thorough description of the algorithm and implementation details, we refer the reader to [Torralba et al., 2017].

The process terminates when all reachable states have been generated or the cost bound b is reached. The result is a BDD, $\beta_\tau = \bigvee_{g\in[0,\dots,b]} \beta_g$, which represents the set of states reachable $\mathcal{S}^r$ in τ with cost bound b. Additionally, we can terminate earlier whenever we reach a state satisfying $G^{\text{hard}} \cup G^{\text{soft}}$. In that case, no lattice search is needed as all soft goals can be reached simultaneously and therefore the set of MUGS is empty.

After computing β_τ, each test for some $G \subseteq G^{\text{soft}}$ within the goal-lattice search (line 8 in Algorithm 2) can be efficiently performed in time polynomial in the size of β_τ and the number of tested goals $|G^{\text{hard}} \cup G|$. This test requires determining whether there exists some state $s \in \mathcal{S}^r$ such that $G^{\text{hard}} \cup G \subseteq s$.

Algorithm 4 Symbolic TestSolvable

1: **Input:** Goal: $G^{\text{hard}} \cup G$, b-reachable states: β_τ
2: **function** SYMBOLICTESTSOLVABLE($G^{\text{hard}} \cup G, \beta_\tau$)
3: $\quad \beta_G \leftarrow \bigwedge_{g\in G^{\text{hard}}\cup G} g$
4: $\quad$ **return** $\beta_\tau \wedge \beta_G \neq \emptyset$

Algorithm 4 shows the simple procedure. First, we produce the BDD β_G representing $\bigwedge_{g\in G\cup G^{\text{hard}}} g$. As this BDD is simply a conjunction of atomic propositions, this only takes

time $O(|\beta_G|)$ where $|\beta_G|$ is the number of nodes in the resulting BDD. This is $O(|G \cup G^{\text{hard}}| \cdot \max_{v \in V} \log_2(|D_v|))$ as each goal over variable v is represented with at most $\log_2(|D_v|) + 2$ BDD nodes.

β_G represents the set of states where the goal is satisfied, $\{s \mid G \cup G^{\text{hard}} \subseteq s\}$, regardless of whether they are reachable or not. Then, the conjunction $\beta_\tau \wedge \beta_G$ results in a BDD that represents the intersection of both sets of states, i.e., the set of states that are reachable and satisfy the goal. Then, G is solvable if and only if such a set is not empty. The conjunction of two BDDs can be computed in time $O(|\beta_\tau||\beta_G|)$, and testing whether a BDD represents the empty set can be done in constant time. Thus, the overall algorithm runs in $O(|G \cup G^{\text{hard}}| \cdot \max_{v \in V} \log_2(|D_v|) \cdot |\beta_\tau|)$.

4.2.2 Goal-Subset Branch and Bound Search

An alternative to symbolic search for state space exhaustion is an exhaustive explicit state space exploration. This approach enables the computation of MUGS via the maximal solvable goal subsets (MSGS). The MSGS refer to the subset maximal soft goal subsets that are satisfied by all reachable goal states.

Proposition 6: $\mathcal{G}^{\text{MSGS}}$ from Exhaustive State Space Exploration

Let τ be an OSP task with soft goals G^{soft} and state space $\Theta_\tau = (\mathcal{S}, L, c, T, I, \mathcal{S}_G, b)$. The maximal solvable goal subsets of τ ($\mathcal{G}^{\text{MSGS}}(\tau)$) are given by $\mathcal{G}^{\text{MSGS}}(\Theta_\tau) := \max_{\subseteq}\{\bigcup_{s \in \mathcal{S}^r_G}(s \cap G^{\text{soft}})\}$, where $\mathcal{S}^r_G \subseteq \mathcal{S}_G$ are all the reachable goal states.

Proof:

$\max_{\subseteq}\{\bigcup_{s \in \mathcal{S}^r_G}(s \cap G^{\text{soft}})\} \subseteq \mathcal{G}^{\text{MSGS}}(\tau)$:

Let $G \in \max_{\subseteq}\{\bigcup_{s \in \mathcal{S}^r_G}(s \cap G^{\text{soft}})\}$. G is solvable because there is a state $s \in \mathcal{S}^r_G$ such that $G \subseteq s$. G is maximal because if there would be a G' with $G \subsetneq G' \subseteq G^{\text{soft}}$ that is solvable and thus $G' \in \bigcup_{s \in \mathcal{S}^r_G}(s \cap G^{\text{soft}})$, G would be removed by $\max_{\subseteq}$.

$\max_{\subseteq}\{\bigcup_{s \in \mathcal{S}^r_G}(s \cap G^{\text{soft}})\} \supseteq \mathcal{G}^{\text{MSGS}}(\tau)$:

Let $G \in \mathcal{G}^{\text{MSGS}}(\tau)$. $G \subseteq G^{\text{soft}}$ is solvable and thus there exists a reachable goal state $s \in \mathcal{S}^r_G$, such that $G \subseteq s$. It follows $G \in \bigcup_{s \in \mathcal{S}^r_G}(s \cap G^{\text{soft}})$. G is not removed by $\max_{\subseteq}$, because it is maximal solvable in τ. This means there is no G' with $G \subsetneq G' \subseteq G^{\text{soft}}$ that is solvable. Thus, there is no reachable goal state $s \in \mathcal{S}^r_G$ where $G' \subseteq s$. It follows $G' \notin \bigcup_{s \in \mathcal{S}^r_G}(s \cap G^{\text{soft}})$.

□

Based on Proposition 6 we introduce a branch-and-bound approach to compute $\mathcal{G}^{\text{MSGS}}(\tau)$. During the exploration of the search space, for each goal state $s_g \supseteq G^{\text{hard}}$ the satisfied soft goals $G^{\text{soft}} \cap s_g$ are tracked. To prune states that cannot further improve the solvable goal subsets, we introduce the concept of a new-goal-subset heuristic.

In classical planning, where the objective is to find a plan, that satisfies all goals, the

heuristic is normally a function $h : S \mapsto \mathbb{R}_0^+ \cup \{\infty\}$, estimating the optimal plan cost. In our case, however, we are looking for all subsets of soft goals that can be satisfied while satisfying G^{hard}. This involves exploring the state space beyond the first goal state, to identify *new* soft goal subsets that have not yet been satisfied. A heuristic can help to prune states that do not lead to new soft goal subsets within the cost bound b. Since *new* is relative to what has already been seen, the needed heuristic does not only depend on the state s and the remaining cost but also on the already satisfied soft-goal subsets.

DEFINITION 21: NEW-GOAL-SUBSET HEURISTIC

Let τ be an OSP task. A **new-goal-subset heuristic** is a function $H : \mathcal{S} \times \mathbb{R}_0^+ \times \mathcal{P}(\mathcal{P}(G^{\text{soft}})) \mapsto \mathbb{R}_0^+ \cup \{\infty\}$.

DEFINITION 22: NEW-GOAL-SUBSET COST

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task with states $\mathcal{S}$. For a state $s \in \mathcal{S}$, remaining cost $b_r \in \mathbb{R}_0^+$ and soft goal subsets $\mathcal{G} \subseteq \mathcal{P}(G^{\text{soft}})$, the **new-goal-subset cost** is

$$\begin{aligned}\min(&\\ &\{c \mid G_{new} \subseteq G^{\text{soft}}, \forall G' \in \mathcal{G} : G_{new} \not\subseteq G',\\ &c = h^*(s, G^{\text{hard}} \cup G_{new}), c \leq b_r\}\\ &\cup \{\infty\}\\ )&\end{aligned}$$

where $h^*(s, G^{\text{hard}} \cup G_{new})$ is the optimal plan cost from s in task $\tau' = (V, A, c, I, G^{\text{hard}} \cup G_{new})$.
The **perfect** goal-subset heuristic, denoted by H^*, assigns every state $s \in \mathcal{S}$ for each remaining cost $b_r \in \mathbb{R}_0^+$ and each $\mathcal{G} \subseteq \mathcal{P}(G^{\text{soft}})$ its new-goal-subset cost.

The new-goal-subset cost for a state s and a set of soft-goal subsets $\mathcal{G}$ is the cost of an optimal plan from s to a within the remaining cost cheapest-to-reach goal subset G for which there is no superset in $\mathcal{G}$. If the new-goal-subset cost of state s is ∞ there are two possible reasons. First, there may be no plan from s, i. e. G^{hard} is not reachable within the remaining cost. Second, there may be no new reachable soft-goal subset.

For a new-goal-subset heuristic to be used safely for pruning, that is not pruning any states from which the MSGS can be improved, it must be admissible.

DEFINITION 23: ADMISSIBLE

Let τ be an OSP task with soft goals G^{soft} and H a new-goal-subset heuristic.
H is called **admissible** if, for all $s \in \mathcal{S}$, bounds $b \in \mathbb{R}_0^+$ and $\mathcal{G} \subseteq \mathcal{P}(G^{\text{soft}})$,

$$H(s, b, \mathcal{G}) \leq H^*(s, b, \mathcal{G}).$$

Next, we present a branch-and-bound (BnB) algorithm called *goal-subset BnB* (GSBnB) that computes all MSGS. In an OSP setting with goal utilities Domshlak and Mirkis [2015] use a similar approach to compute an optimal plan that maximizes the utility. The pseudocode is given in Algorithm 5.

Algorithm 5 Goal-Subset BnB (GSBnB)

```
1:  Input: OSP task τ, new-goal-subset heuristic H_p, heuristic H_g, under-approximation of
    solvable goal subsets M̃
2:  function GSBnB(τ = (V, A, c, I, G^hard, G^soft, b), H_p, H_g, M̃ = {{}})
3:      G ← M̃                                   ▷ current maximal solvable goal subsets
4:      g(s) ← ∞ for all states s ≠ I and g(I) ← 0
5:      O ← {(I, 0)}                  ▷ priority queue ordered by ascending H_g(s, b − g(s), G)
6:      while |O| > 0 do
7:          s, g_s ← POP(O)
8:          if g(s) < g_s then                                          ▷ duplicate check
9:              continue
10:         if G^hard ∪ G^soft ⊆ s then
11:             return {G^soft}                                         ▷ all goals are solvable
12:         if G^hard ⊆ s then
13:             UPDATE(G, s ∩ G^soft)
14:         for all a ∈ A(s) do
15:             s' ← s⟦a⟧
16:             g_s' ← g(s) + c(a)
17:             if g_s' ≥ g(s') then                                    ▷ duplicate check
18:                 continue
19:             g(s') ← g_s'
20:             if g(s') > b ∨ H_p(s, b − g(s'), G) = ∞ then
21:                 continue                          ▷ prune s' if nothing new reachable
22:             O ← O ∪ (s', g(s'))
23:     return G
24: function UPDATE(G, G)
25:     if ∀G' ∈ G : G ⊈ G' then
26:         G ← (G \ {G' ∈ G | G' ⊊ G}) ∪ {G}
```

$\mathcal{G}$ stores the currently maximal solvable goal subsets (line 3). If the hard goals are satisfied, then $\mathcal{G}$ is updated with the satisfied soft goals $G = s \cap G^{\text{soft}}$ (line 13). If G is not a subset of any set in $\mathcal{G}$, then G is added and all subsets of G are removed. This ensures that $\mathcal{G}$ always contains the *maximal* solvable goal subsets. If all hard and soft goals are satisfied the algorithm terminates early (line 10), because then all soft goals are solvable. Duplicate checking and reopening of states reached via a cheaper path are handled as in A^* [Hart et al., 1968].

The new-goal-subset heuristic is used to prune states, from which no new soft goal subset can be reached within the cost bound (line 20). Given that H_p does not only depend on the state s and the remaining cost but also on the soft goal subsets $\mathcal{G}$ that have been reached, the heuristic value can change between the time a state is added to the open list (line 22) and when it is removed from the open list to be expanded (line 7). Thus, it could make sense to reevaluate H_p if $\mathcal{G}$ has changed between these two points. We leave this extension for future work, and evaluate H_p for each state only before it is inserted into the open list.

If an admissible new-goal-subset heuristic is used for pruning, then GSBnB solves allMUGS by computing all MSGS.

PROPOSITION 7: CORRECTNESS GSBNB

Given an OSP task τ and an admissible new-goal-subset heuristic H_p, $GSBnB(\tau, H_p, H_g)$ computes $\mathcal{G}^{\text{MSGS}}(\tau)$.

Proof:
To prove that $\mathcal{G} = \mathcal{G}^{\text{MSGS}}(\tau)$ at the end of Algorithm 5, we have to show, that (1) for all $G \in \mathcal{G}^{\text{MSGS}}(\tau)$ GSBnB visits a state s such that $G = s \cap G^{\text{soft}}$, (2) no state s with $G = G^{\text{soft}} \cap s$ for which exists $G' \in \mathcal{G}^{\text{MSGS}}(\tau)$ such that $G' \subsetneq G$ is reached, and (3) $\mathcal{G}$ only contains subset maximal sets.

(1) GSBnB only terminates, if $G^{\text{hard}} \cup G^{\text{soft}}$ is reached or if all states that are not pruned due to (a) a duplicate check, (b) exceeding of cost bound $g(s) > b$ or (c) pruning based on H_p, ($H_p(s, b - g(s'), \mathcal{G}) = \infty$), are explored. In the first case, a plan π with $cost(\pi) \leq b$ satisfying $G^{\text{hard}} \cup G^{\text{soft}}$ is found and thus $\{G^{\text{soft}}\} = \mathcal{G}^{\text{MSGS}}(\tau)$ is returned. For the latter case, we address each condition separately.

   (a) If a state s with g_s is pruned due to (a) a duplicate check, then a shorter path to s with cost g'_s has been found. Since all states reachable from s with remaining cost $b - g'_s$ can also be reached with remaining cost $b - g_s$, state s reached via a more expensive path can be pruned.

   (b) If (b) the cost to reach a state s exceeds the cost bound $g(s) > b$, then s is not reachable. If $g(s)$ is not the minimal cost to reach s, then pruning based on (a) duplicate check will not prune s if it is reached via cheaper paths and thus s is considered later if it is reachable within cost bound b.

   (c) If a state s is pruned (c) based on H_p and a $G^{\text{hard}} \cup G$ with $G \in \mathcal{G}^{\text{MSGS}}(\tau)$ is reachable from s within the cost bound, then this means G must be in $\mathcal{G}$. Otherwise because H_p is admissible and $G^{\text{hard}} \cup G$ reachable, we have $H_p(s, b - g(s), \mathcal{G}) \leq H^*(s, b - g(s), \mathcal{G}) \leq h^*(s, G^{\text{hard}} \cup G) < \infty$.

(2) holds because of Proposition 6 and the fact that only states reachable within the cost bound b are explored.

(3) holds because all subsets are removed by the function UPDATE.

□

In contrast to A^*, the order in which states are popped from the open list does not affect the correctness of the solution (Proposition 7). However, it can affect the performance. Guiding the search first to states that promise to solve many goals could have a positive effect on the pruning function, as $\mathcal{G}$ grows quickly at the beginning, which could then lead to more effective pruning. We evaluate this hypothesis empirically in Section 4.3.1.

Note that, by computing $\mathcal{G}^{\text{MSGS}}(\tau)$, the algorithm also computes an optimal plan to a state that satisfies the maximum number of soft goals. In Section 4.3.1 we compare our approach to the state of the art in OSP planning.

IMPLEMENTATION OF GOAL-SUBSET HEURISTICS

The new-goal-subset cost is defined as the minimal cost of a plan for any new reachable soft goal subset. However, when computing the new-goal-subset cost it is not necessary to consider every new soft goal subset, but only the subset minimal goal subsets. All larger goal subsets are at least as expensive and are dominated in the minimization step. We refer to the set of all minimal new soft goal subsets with respect to $\mathcal{G}$ as $\mathcal{G}^{min}_{new} = \{G_{new} \subseteq G^{\text{soft}} \mid \forall G \in \mathcal{G}, \exists g \in G^{\text{soft}} : G_{new} \setminus G = \{g\}\}$. A straightforward method for computing $H(s, b, \mathcal{G})$ involves iterating over $\mathcal{G}^{min}_{new}$ and computing $h(s, G^{\text{hard}} \cup G_{new})$ for each $G_{new} \in \mathcal{G}^{min}_{new}$. However, $\mathcal{G}^{min}_{new}$ can become large, and each call to h can be computationally expensive. Given that the heuristic is computed for every state during the search, implementing this approach is bound to cause a substantial overhead. Thus, we introduce an approximation of the new-goal-subset cost using classical planning heuristics of singleton goals.

DEFINITION 24: NEW-GOAL-SUBSET COST ESTIMATION WITH SINGLETON GOALS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and $\eta = \{h_g \mid g \in G^{\text{hard}} \cup G^{\text{soft}}\}$ a set of admissible classical planning heuristics for tasks $\tau_g = (V, A, c, I, \{g\})$. The estimation of H^* for $s \in \mathcal{S}$, remaining cost $b_r \in \mathbb{R}^+_0$ and soft goal subsets $\mathcal{G} \subseteq \mathcal{P}(G^{\text{soft}})$ based on singleton goals is defined as

$$H_\eta(s, b, \mathcal{G}) = \begin{cases} \infty & \text{if } \exists g \in G^{\text{hard}} : h_g(s) > b_r \\ \infty & \text{if } \exists G \in \mathcal{G} : G_{\leq b} \subseteq G \\ 0 & \text{otherwise} \end{cases}$$

where $G_{\leq b_r} = \{g \in G^{\text{soft}} \mid h_g(s) \leq b_r\}$.

The estimation based on singleton goal facts considers the hard and soft goals separately. Since all hard goals must be satisfied, if one hard goal estimate exceeds the remaining cost then, G^{hard} cannot be satisfied, and thus the state can be pruned. As enumerating all new soft goal subsets and estimating which ones are reachable within the cost bound is not feasible, we take the opposite approach. We check if an over-approximation of the

reachable soft goals $G_{\leq b}$ is *new*, i. e. no super set of $G_{\leq b}$ is contained in $\mathcal{G}$. $G_{\leq b}$ contains all soft goals, that are individually reachable within the remaining cost b_r. If we have already seen a superset of $G_{\leq b}$, then we know that no new soft goal subsets are reachable, and the state can be pruned. In case all hard goals are reachable and the reachable soft goals are new, we do not want to prune the state and thus return 0. Some example heuristic evaluations are given in Example 9.

EXAMPLE 9: NEW-GOAL-SUBSET COST ESTIMATION WITH SINGLETON GOALS

Let's consider the task τ with $G^{\text{hard}} = \emptyset$ and $G^{\text{soft}} = \{A, B, C, D, E\}$. Let $\mathcal{G} = \{\{A, B\}, \{B, C\}, \{C, D, E\}\}$ be the current maximal solvable goal subsets and $b_r = 3$ the remaining available cost.

s_1: with the heuristic estimates:

G^{soft}	A	B	C	D	E
h_g	3	2	4	1	4

$G_{\leq 3} = \{A, B, D\} \rightarrow H_\eta(s_1, 3, \mathcal{G}) = 0$

Thus, s_1 is not pruned, because no superset of $\{A, B, D\}$ has been reached and thus $\{A, B, D\}$ could improve the MSGS.

s_2: with the heuristic estimates:

G^{soft}	A	B	C	D	E
h_g	4	2	1	4	4

$G_{\leq 3} = \{B, C\} \rightarrow H_\eta(s_2, 3, \mathcal{G}) = \infty$

Thus, s_2 is pruned, because $\{B, C\}$ has already been reached.

To ensure the safe use of H_η for pruning, it needs to be admissible (Proposition 7). This is the case if only admissible estimates are used for the individual goals.

PROPOSITION 8: ESG IS AN ADMISSIBLE GOAL-SUBSET HEURISTIC

Given OSP $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ and $\eta = \{h_g \mid g \in G^{\text{hard}} \cup G^{\text{soft}}\}$ a set of *admissible* classical planning heuristics for tasks $\tau_g = (V, A, c, I, \{g\})$, H_η is an admissible goal-subset heuristic.

Proof:
We consider each case of $H_\eta(s, b, \mathcal{G})$ separately. Recall $G_{new} \subseteq G^{\text{soft}}$ and for all $G \in \mathcal{G} : G_{new} \not\subseteq G$. (1) If $G \subseteq G'$ then $h^*(s, G) \leq h^*(s, G')$ since all plans satisfying G also satisfy G'.

(a) If $\exists g \in G^{\text{hard}} : h_g(s) > b$ then $H_\eta(s, b, \mathcal{G}) = \infty$. Let $g \in G^{\text{hard}}$ such that $h_g(s) > b$. Since h_g is admissible, i. e. $h_g(s) \leq h^*(s, \{g\})$ and (1) we have $b < h_g(s) \leq$

$h^*(s, \{g\}) \leq h^*(s, G^{\text{hard}} \cup G)$ for any $G \subseteq G^{\text{soft}}$ and thus $H^*(s, b, \mathcal{G}) = \infty$.

(b) If $\exists G \in \mathcal{G} : G_{\leq b} \subseteq G$ then $H_\eta(s, b, \mathcal{G}) = \infty$. Let $G \in \mathcal{G}$ such that $G_{\leq b} \subseteq G$. Because $G \in \mathcal{G}$ for all G_{new}, $G_{new} \nsubseteq G$ holds, and thus $(G^{\text{soft}} \setminus G_{\leq b}) \cap G_{new} \neq \emptyset$. Let $g \in (G^{\text{soft}} \setminus G_{\leq b}) \cap G_{new}$. For all $g' \in G^{\text{soft}} \setminus G_{\leq b}$, because $h_{g'}$ is admissible we have $b < h_{g'} < h^*(s, \{g'\})$. Then from (1) it follows that $b < h_{g'} < h^*(s, \{g'\}) < h^*(s, G^{\text{hard}} \cup G_{new})$ for all G_{new}. Thus, $H^*(s, b, \mathcal{G}) = \infty$.

(c) Otherwise, $H_\eta(s, b, \mathcal{G}) = 0$ which is admissible, because $H^*(s, b, \mathcal{G}) \geq 0$ for any input.

□

In general, any admissible heuristic can be used to compute the heuristic for each goal fact $g \in G^{\text{hard}} \cup G^{\text{soft}}$ by computing the heuristic for task $\tau_g = (V, A, c, I, \{g\})$. However, this requires $|G^{\text{hard}} \cup G^{\text{soft}}|$ computations, which can be excessive. To compute h_g for each goal efficiently, we consider heuristic functions that are either quickly to evaluate or that estimate the cost of reaching a state that satisfies a goal G by calculating an admissible estimate for each goal fact $g \in G$ individually. The latter requires only one heuristic evaluation per H_η computation.

Max Heuristic h^{max} [Bonet and Geffner, 2001] approximates the remaining cost by the maximal cost of reaching each goal fact individually (see Definition 7). The last step in the h^{max} computation is $\max_{g \in G} h^{\text{max}}(s, \{g\})$. This allows to extract, from a single h^{max} computation for task $\tau' = (V, A, c, I, G^{\text{hard}} \cup G^{\text{soft}})$, for each goal fact $g \in G^{\text{hard}} \cup G^{\text{soft}}$ the estimation $h^{max}_g(s) \coloneqq h^{\text{max}}(s, \{g\})$.

Abstraction Heuristic Abstraction heuristics [Helmert et al., 2007] use the perfect heuristic in the abstract state space to estimate the remaining cost in the concrete state space (see Definition 9). Often multiple abstractions and a cost partitioning [Seipp et al., 2017] are used to achieve a compatible performance. Here we are using one cartesian abstraction [Seipp and Helmert, 2018] for each goal fact $g \in G^{\text{hard}} \cup G^{\text{soft}}$. Each abstraction α_g is based on the task $\tau_g = (V, A, c_\tau, I_\tau, \{g\})$. This corresponds to the abstraction by goal strategy described by Seipp and Helmert [2018]. To generate an abstraction α_g, counter-example guided abstraction refinement, as introduced by Seipp and Helmert [2018], is used. The heuristic estimate for each goal fact is then given by $h^{car}_g(s) \coloneqq h^*_{\Theta^{\alpha_g}}(\alpha_g(s))$.

Potential Heuristics Potential heuristics [Pommerening et al., 2015] assign a numeric value to each fact called *potential*. The sum of the potentials of all facts satisfied by state s is then used as heuristic estimate (see Definition 10). Thus, the evaluation of potential heuristics is quite fast. Here, for each task $\tau_g = (V, A, c, I, \{g\})$ with $g \in G^{\text{hard}} \cup G^{\text{soft}}$ we use one potential heuristic h^{pot}_g.

4.3 Computational Evaluation

First, we evaluate the performance of the presented algorithms on simple soft goal atoms based on an OSP version of the IPC domains. Then, we analyzed the scalability of our approach for more complex soft goals by utilizing compilations of temporal goals to goal facts.

4.3.1 Soft-Goal Atoms

Experiment Setup & Benchmarks

Our implementation is based on the planning framework Fast Downward[2] (FD) [Helmert, 2006]. We used the following search algorithms, which are either included in the official FD release or are available for public use.

- *Greedy Best First Search* (GBFS): described in [Doran and Michie, 1966] and provided by FD
- *Symbolic Search* (Sym): described by Torralba et al. [2017] and implemented in symbolic Fast Downward[3].

As heuristic functions, which are either included in the official FD release or are available for public use we used the following implementations and specifications:

- $blind$: It returns 0 for goal states and the minimal action cost otherwise. It is used as a baseline for the exhaustive state space search.
- h^{FF} [Hoffmann and Nebel, 2001]: It is a non-admissible heuristic based on delete-relaxation. We used the implementation provided by FD.
- h^{max} [Bonet and Geffner, 2001]: We used the implementation provided by FD.
- h^{pot} [Seipp et al., 2015]: We used the implementation of potential heuristics provided by FD, with one potential function per goal fact optimized each for 1000 sample states.
- h^{car} [Seipp and Helmert, 2018]: We used the implementation for Cartesian abstractions generated with counter example guided abstraction refinement provided by FD, with the default configuration (maximum number of abstract states overall: ∞, maximum number of transition overall: 1M, maximum time for abstraction generation: ∞, split selection strategy: max refined). Using the goals subtasks generator generates one abstraction per goal state. We modified the implementation to provide a heuristic value of 0 for each goal fact satisfied in the initial state and did not use any cost partitioning, i. e. the sum of the estimates are not admissible but the estimate for each goal fact is.

We ran the following combinations of (meta-)search and heuristics:

[2] `https://github.com/aibasel/downward`
[3] `https://gitlab.com/atorralba/fast-downward-symbolic`

- Goal Lattice Search (GLS)
  - expansion orders:
    - systematic weakening (SysW)
    - systematic strengthening (SysS)
  - solvability check:
    - explicit cost-bounded search: GBFS with h^{FF} (GBFS(h^{FF}))
    - symbolic cost-bounded forward search (Sym)
- Goal-Subset BnB (GSBnB(h_p, h_g)), if the same heuristic is used for pruning and guidance we denote this by GSBnB(h).
  - with pruning heuristic h_p based on:
    - $blind$ (baseline, no pruning)
    - h^{max}
    - h^{pot}
    - h^{car}
  - with guidance heuristic h_g based on:
    - $blind$ (baseline)
    - $h_p = h_g$: sum of the individual goal estimates
    - h^{FF}

The extensions of Fast Downward and symbolic Fast Downward are publicly available[4].

All experiments are performed on a cluster with Intel E5-2695 v4 2.1G machines. If not mentioned differently the time-out it set to 30 min and the memory limit to 4 GB.

Benchmarks As benchmarks, we used the instances from the International Planning Competition (IPC) [classical domains, 2018], in particular the instances from the optimal track. Those are not OSP tasks but rather classical planning tasks. This means they have no cost bound b and only one set of goals G. To obtain OSP domains, we follow [Domshlak and Mirkis, 2015, Katz et al., 2019] and use all goals as soft goals $G^{\mathsf{soft}} = G$ and no hard goals $G^{\mathsf{hard}} = \emptyset$ and as cost bound $b = x * c$ where $x \in \{0.25, 0.5, 0.75\}$ and c is the best known solution cost [api.planning.domains, 2018]. The specific cost bounds we used are publicly available[5]. Due to an implementation restriction, we only consider instances with less than 32 goal facts.

In the following, we take a closer look at the most important features of the benchmark instances, the number of soft goals and the number and size of the MUGS. Figure 8 depicts the number of goal facts for the tasks in the IPC benchmark set. Many domains scale difficulty with the number of goal facts. For example Miconic, which models an elevator controller, scales by increasing the number of served passengers. Other domains like Agricola, FreeCell, Movie, Pegsol, Termes and Tidybot have a fixed number of goal facts,

[4] https://doi.org/10.5281/zenodo.14989835
[5] https://doi.org/10.5281/zenodo.14988342

due to the structure of the problem they model. In FreeCell for example, there are four goals one for each color of playing cards. Agricola has only one goal indicating that the last stage of a harvesting game has been completed successfully. In Section 4.3.1 we will analyze how the different approaches scale with respect to the number of soft goals.

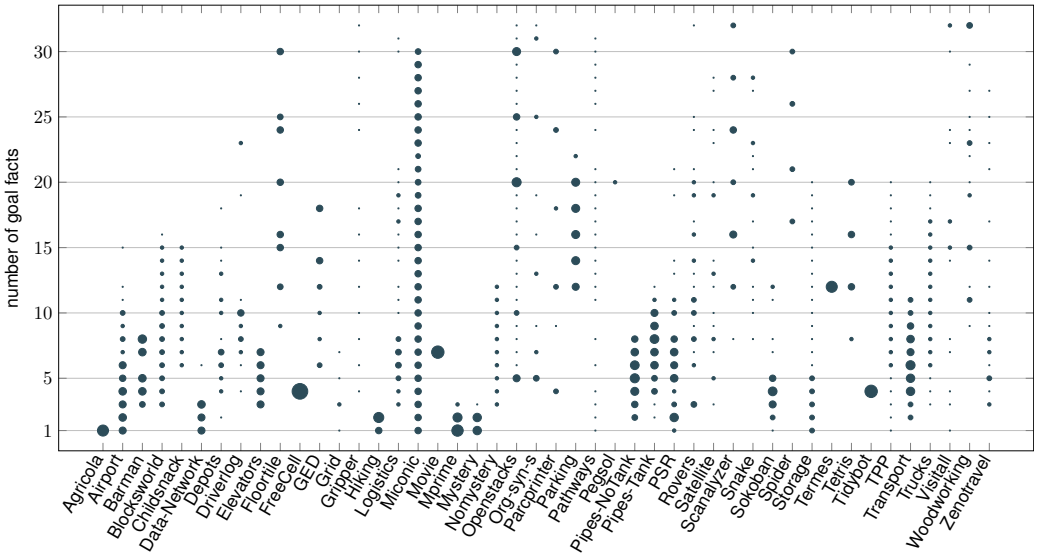


Figure 8: Distribution of number of goal facts in the IPC domains. The larger the dot, the more instances with the corresponding number of goal facts exist.

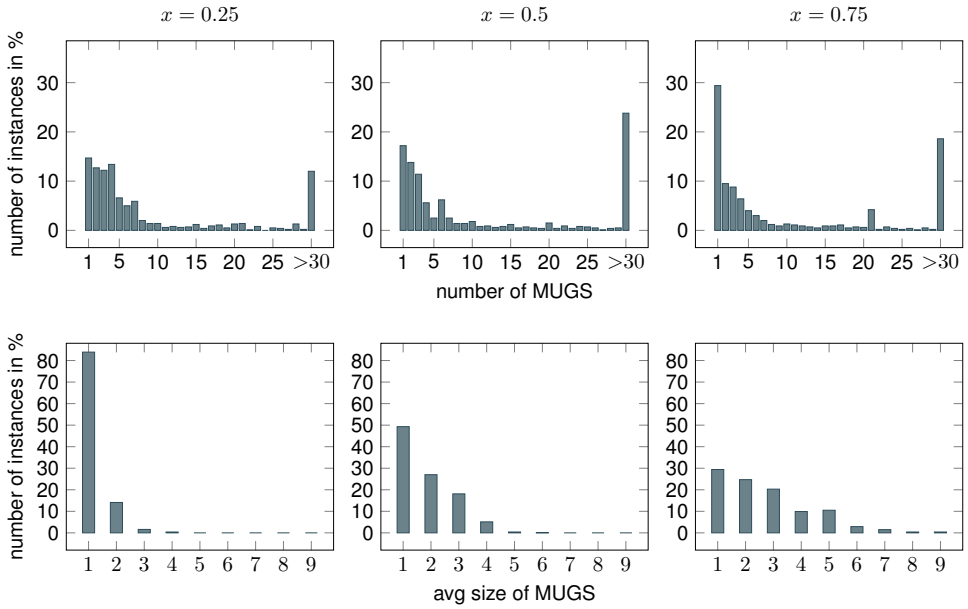


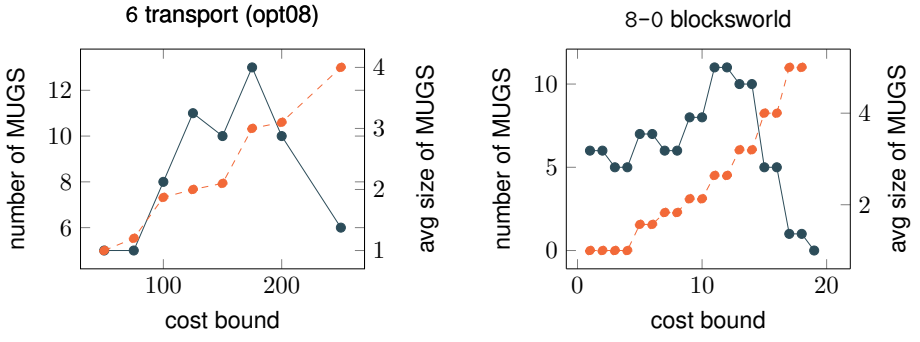
Figure 9: Distribution of number (top) and average size (bottom) of MUGS over commonly solved instances among all cost bounds of the IPC OSP benchmark set. For the size only, instances with less than 100 MUGS are considered. From left to right cost bound of 0.25, 0.5 and 0.75 times best known solution cost. All instances with more than 30 MUGS are aggregated at > 30.

The objective of our algorithms is to compute allMUGS. Figure 9 depicts the distribution of number and size of MUGS for all instances, of which one of our algorithms could compute all MUGS. We only consider instances where MUGS are known for all cost bounds, which allows us to analyze the number and size changes by increasingly weaker bounds. For the size, we only consider data for instances with less than 100 MUGS. For a very tight bound $x = 0.25$ the number of MUGS resembles more or less the distribution of goal facts, as most of them have size one. This indicates that, in many domains, individual goal facts share common actions in their plans. Therefore, if the cost bound is tight, achieving even a single goal fact is not feasible. With increasing cost bound, the number of MUGS decreases and their size increases. 50% of the instances have less than 6, 7 and 6 MUGS for the cost bounds 0.25, 0.5 and 0.75 respectively. The average size of MUGS stays below 10 for almost all tasks. For cost bounds $0.25/0.5/0.75$ there are $140/165/119$ instances with more than 100 MUGS. The maximum is $74613/48620/26334$ for a gripper instance. On average there are $222/530/361$ (arithmetic mean) and $9.3/11.8/7.8$ (geometric mean) MUGS per instance.

As the number of goal fact varies per domain and instance, the number and size of MUGS also varies. The problem structure itself influences how the number and size of MUGS changes when the cost bound is changed. Generally, the number of MUGS begins at the number of goal facts $|G|$, when the cost bound is so low, that no single goal fact can be satisfied, and ends at 0 when the cost bound is high enough to satisfy all goal facts (assuming this is possible). The number of MUGS can vary between these two extremes. For a common pattern see Example 10. The evaluation of the number of MUGS and their size, for each domain can be found in Appendix A.1.

EXAMPLE 10: MUGS WITH INCREASING COST BOUND

For tasks with independent goals – that is, goals that can be satisfied separately from others (e.g., transport, where each package can be delivered individually) – the number of MUGS first increases as more combinations of goal facts can be satisfied. For example different combinations of packages can be delivered. At some point, the number of MUGS decreases again as more goal facts are satisfied together, leading to the merging of MUGS. The average size begins at 1 and increases with rising cost bound. The following graphs depict the number (blue/solid) and size (orange/dashed) of MUGS for an increasing cost bound in a Transport and Blocksworld instance:



OVERALL COVERAGE

An instance is considered solved if all MUGS could be computed within the specified time and memory limit. We first evaluate the two main approaches, goal-lattice (GLS) and branch-and-bound search (GSBnB), individually and then compare the best performing configurations to each other.

The coverage per domain for GLS is presented in Table 1. The symbolic approach for testing solvability performs significantly better than the solvability test based on explicit state space search. It solves on average more than 200 tasks more. Overall, there are only three domains where explicit search performs better: Airport, Org-syn-s and Sokoban. In the following we will focus on GLS with symbolic search and will refer to it just as SysW and SysS. As expected, for lower cost bounds, 0.25 and 0.5, SysW performs better, while for 0.75 SysS solves more instances.

Next, we analyze the coverage for the branch-and-bound search (GSBnB) approach, considering the different heuristics for pruning and for guidance. The coverage results for all configurations are given in Table 2. First we focus on the impact of the guidance heuristic. The order in which states are visited, has an impact on the pruning. If states that satisfy many soft goals are included early on in the current MSGS, then the bar for *new* subsets is higher, and therefore more states are pruned. For a cost bound of 0.25, the use of a guidance heuristic decreases the coverage. However, for larger cost bounds, additional guidance has a positive effect. Overall, it is not worthwhile to compute h^{FF} in addition to the heuristic used for pruning.

Next we assess the impact of the pruning function and the chosen heuristic. In comparison with the baseline to no pruning, all pruning heuristics have a positive effect, on average, increasing the coverage by about 30 instances. The greatest impact is observed for cost bound 0.5. There is no universally best choice; the most suitable heuristic varies depending on the cost bound. For cost bounds 0.25 the best choice is GSBnB$(h^{\mathsf{car}}, blind)$, for cost bound 0.5 GSBnB(h^{pot}) and for cost bound 0.75 it is GSBnB(h^{car}). GSBnB(h^{car}) has the most consistent performance, while h^{pot} only has a higher overall coverage for 0.5 due to the single domain FreeCell.

Finally, we compare the best approaches from GLS and GSBnB to each other. If we just consider the overall coverage, then GSBnB performs best. When considering the most effective approach for each cost bound, GSBnB solves 36, 44 and 9 instances more for each cost bound respectively. However, comparing the number of domains in which each approach solves the most instances, as shown in Figure 3, no clear winner emerges. While GSBnB performs better at cost bounds 0.25 and 0.5, the number of domains where each approach performs better is almost equal at cost bound 0.75.

Comparing the different pruning heuristics for GSBnB, shows that, except for 0.25, h^{car} performs better in more domains than h^{max} or h^{pot}.

SCALING OVER SOFT-GOAL COMPLEXITY

To determine when GSBnB and GLSperform better, we take a closer look at the influence of the number of soft goals. The relative coverage as a function of the number of soft goals is shown in Figure 10.

	$x = 0.25$				$x = 0.5$				$x = 0.75$			
	SysW		SysS		SysW		SysS		SysW		SysS	
domain	h^{FF}	Sym	h^{FF}	Sym	h^{FF}	Sym	h^{FF}	Sym	h^{FF}	Sym	h^{FF}	Sym
Agricola(20)	**20**	**20**	**20**	**20**	15	**20**	16	**20**	3	**20**	3	**20**
Airport(50)	24	23	**25**	23	19	21	**22**	21	20	19	**21**	18
Barman(34)	13	**24**	15	**24**	4	**13**	4	**13**	4	**11**	5	**11**
Blocksworld(35)	29	32	**35**	**35**	21	**29**	26	**29**	17	**23**	18	**23**
Childsnack(20)	0	**6**	0	**6**	0	**4**	0	**4**	0	**4**	0	**4**
Data-Network(20)	17	**19**	**19**	**19**	14	**17**	14	**17**	11	**13**	12	**13**
Depots(22)	12	**15**	14	**15**	7	**11**	7	9	4	**7**	4	**7**
Driverlog(18)	14	**15**	**15**	**15**	10	**14**	12	**14**	8	**12**	9	**12**
Elevators(30)	26	**30**	28	**30**	22	**25**	24	**25**	17	**24**	20	**24**
Floortile(36)	6	11	10	**18**	2	**11**	2	9	2	**6**	2	4
FreeCell(80)	36	**76**	44	**76**	15	**29**	19	**29**	14	**19**	14	18
GED(20)	15	15	**20**	**20**	10	15	15	**19**	10	**15**	13	**15**
Grid(5)	4	**5**	4	**5**	**3**	**3**	**3**	**3**	**2**	**2**	**2**	**2**
Gripper(14)	5	6	6	**10**	4	**8**	4	**8**	4	**10**	4	7
Hiking(20)	14	**19**	15	**19**	11	16	12	**17**	11	**15**	11	**15**
Logistics(61)	22	37	26	**38**	14	**26**	18	**26**	12	**23**	12	**23**
Miconic(150)	55	75	69	**110**	45	**90**	47	89	41	**94**	45	77
Movie(30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
Mprime(35)	**35**	**35**	**35**	**35**	29	**34**	28	**34**	24	28	23	**29**
Mystery(19)	**19**	**19**	**19**	**19**	18	**19**	18	**19**	16	**17**	16	**17**
Nomystery(20)	16	**20**	**20**	**20**	10	**16**	10	**16**	8	**12**	8	**12**
Openstacks(58)	17	17	29	**38**	15	17	29	**38**	15	19	29	**38**
Org-syn-s(15)	7	8	**10**	8	7	8	**9**	8	6	**8**	**8**	**8**
Parcprinter(15)	7	7	**9**	**9**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
Parking(40)	5	14	16	**22**	0	**2**	1	**2**	**0**	**0**	**0**	**0**
Pathways(24)	5	**7**	5	**7**	4	**5**	4	**5**	4	**5**	4	**5**
Pegsol(2)	0	0	**1**	**1**	0	0	**1**	**1**	**0**	**0**	**0**	**0**
Pipes-NoTank(50)	38	**45**	43	**45**	20	**29**	24	**29**	16	**22**	18	**22**
Pipes-Tank(50)	21	**41**	27	**41**	13	**24**	16	**24**	9	**18**	10	**18**
PSR(50)	**48**	**48**	**48**	**48**	**48**	**48**	**48**	**48**	**48**	**48**	**48**	**48**
Rovers(31)	12	**19**	12	**19**	7	**14**	7	**14**	6	**14**	6	**14**
Satellite(19)	7	12	8	**15**	6	**12**	6	**12**	4	**7**	5	**7**
Scanalyzer(22)	4	4	10	**11**	4	**10**	4	**10**	4	**10**	4	4
Snake(17)	5	6	**8**	**8**	3	**8**	6	7	2	**6**	3	**6**
Sokoban(30)	29	**30**	**30**	**30**	26	27	**28**	27	23	23	**25**	23
Spider(12)	0	0	**6**	**6**	0	0	**6**	**6**	0	0	5	**6**
Storage(30)	18	19	18	**20**	16	**17**	16	**17**	15	**16**	15	**16**
Termes(20)	5	**20**	18	**20**	1	**18**	7	**18**	0	**15**	3	**15**
Tetris(17)	7	7	12	**15**	3	8	7	**9**	3	**8**	3	7
Tidybot(30)	29	**30**	**30**	**30**	21	**27**	25	**27**	10	**19**	15	**19**
TPP(30)	8	**12**	8	**12**	6	**8**	6	**8**	6	**8**	6	**8**
Transport(59)	26	**36**	28	**36**	19	**26**	24	**26**	22	**23**	22	**23**
Trucks(30)	9	17	12	**19**	6	**12**	6	**12**	5	**9**	5	**9**
Visitall(13)	6	6	**11**	10	7	8	**10**	**10**	7	**10**	8	9
Woodworking(20)	5	5	**14**	12	5	**8**	7	**8**	5	**8**	5	7
Zenotravel(20)	12	**14**	13	**14**	8	**12**	9	**12**	8	**10**	8	**10**
sum(1443)	742	956	895	**1083**	555	806	644	**836**	483	**717**	534	710

Table 1: Overall coverage for GLS approaches on OSP IPC benchmarks. Cost bounds set to x times the best known plan cost. Best result per cost bound highlighted in **bold**.

	$x = 0.25$								$x = 0.5$								$x = 0.75$							
h_{lp}		$h^{\max}$		h^{pot}		h^{car}			$h^{\max}$		h^{pot}		h^{car}				$h^{\max}$		h^{pot}		h^{car}			
h_g		$h^{\max}$	h^{FF}	h^{pot}	h^{FF}	h^{car}	h^{FF}		$h^{\max}$	h^{FF}	h^{pot}	h^{FF}	h^{car}	h^{FF}			$h^{\max}$	h^{FF}	h^{pot}	h^{FF}	h^{car}	h^{FF}		
Agricola(20)	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	19	17	18	**20**	**20**	19	**20**	**20**	16	11	4	2	4	**17**	**17**	6	11	12	3
Airport(50)	29	31	32	30	29	27	**33**	**33**	25	25	26	26	**27**	24	26	**27**	26	23	24	24	23	24	24	23	**25**	**25**	24
Barman(34)	18	23	21	18	18	18	**25**	**25**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	11	4	**10**	4	4	4	4	4	8	8	6
Blocksworld(35)	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	29	29	**30**	29	29	29	29	29	29	21	22	23	**24**	21	21	21	21	22	22
Childsnack(20)	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Data-Network(20)	18	**20**	**20**	19	19	**20**	**20**	**20**	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**	13	**15**	14	13	13	13	13	**15**	**15**	**15**
Depots(22)	16	17	16	17	17	15	18	17	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
Driverlog(18)	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	11	**12**	**12**	11	11	11	11	11	**12**	**12**
Elevators(30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**25**	**25**	**25**	**25**	**25**	24	**25**	**25**	**25**	24	24	23	25	24	24	24	24	24	**27**
Floortile(36)	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	2	2	**3**	**3**	2	2	2	2	2	2
FreeCell(80)	**80**	**80**	**80**	**80**	**80**	**80**	**80**	72	31	37	39	37	67	**71**	57	36	36	31	21	20	19	20	29	**31**	24	22	22	20
GED(20)	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
Grid(5)	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	3	3	3	3	3	3	3	**4**	**4**	2	**3**	2	3	2	2	**3**	**3**	**3**	**3**
Gripper(14)	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**6**	**6**	**6**	5	**6**	**6**	**6**	5	5	5	5	**6**	**6**	5	**6**	**6**	5	**6**	**6**
Hiking(20)	19	**20**	**20**	**20**	**20**	**20**	**20**	**20**	14	16	16	14	16	16	15	**18**	**18**	13	**14**	13	13	13	13	13	**14**	**14**	**14**
Logistics(61)	29	33	34	33	33	35	**36**	**36**	21	23	23	23	21	21	21	23	**24**	16	18	**19**	**19**	16	18	17	**19**	**19**	**19**
Mficonic(150)	93	93	93	93	91	93	**95**	**95**	64	65	64	64	65	64	63	**66**	65	**55**	**55**	**55**	**55**	**55**	**55**	**55**	**55**	**55**	**55**
Movie(30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
Mprime(35)	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	32	34	34	33	33	32	**35**	**35**	**35**	27	**31**	**31**	**31**	28	28	28	**31**	**31**	**31**
Mystery(19)	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	17	18	18	18	18	18	17	**19**	**19**	**19**
Nomystery(20)	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	14	16	15	15	14	14	14	16	**18**	16	10	10	10	10	10	9	**12**	**12**	**12**
Openstacks(58)	45	45	44	45	45	44	**50**	**50**	40	**41**	40	**41**	40	40	40	40	40	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
Org-syn-s(15)	10	**15**	**15**	13	13	11	10	10	9	**15**	**15**	11	10	10	9	9	9	9	**15**	**15**	9	9	9	9	9	9	9
Parcprinter(15)	**15**	**15**	**15**	**15**	14	**15**	**15**	**15**	13	**14**	**14**	13	13	13	13	**14**	**14**	**12**	**12**	**12**	**12**	**12**	**12**	11	**12**	**12**	**12**
Parking(40)	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**26**	**7**	6	6	**7**	**7**	**7**	**7**	**7**	**7**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Pathways(24)	5	**6**	**6**	**6**	5	**6**	**6**	**6**	4	**5**	**5**	4	4	4	4	**5**	**5**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
Pegsol(2)	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**
Pipes-NoTank(50)	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	33	36	**37**	**37**	36	36	32	36	33	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**
Pipes-Tank(50)	37	36	37	39	**40**	38	38	38	21	21	21	21	**24**	**24**	23	24	22	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**
PSR(50)	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**
Rovers(31)	16	**17**	**17**	**17**	16	16	16	16	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
Satellite(19)	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**
Scanalyzer(22)	13	12	12	**15**	14	13	14	14	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	7	7	7	**10**	**10**	7	**10**	**10**	7
Snake(17)	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**	15	14	14	16	15	14	15	14	14	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Sokoban(30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	26	**28**	**28**	**28**	26	26	24	27	**28**	**28**
Spider(12)	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**11**	**11**	10	10	**11**	**11**	**11**	**11**	**11**	10
Storage(30)	21	21	21	**22**	21	**22**	**22**	**22**	17	**18**	17	**18**	**18**	**18**	**18**	**18**	**18**	15	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**
Termes(20)	**20**	**20**	19	**20**	**20**	**20**	**20**	**20**	**16**	**16**	15	14	**16**	**16**	**16**	**16**	**16**	**12**	**12**	11	10	**12**	11	**12**	**12**	**12**	**12**
Tetris(17)	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**13**	**13**	12	12	**13**	**13**	**13**	**13**	**13**	**11**	9	9	9	10	10	9	**11**	**11**	10
Tidybot(30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**29**	28	28	28	**29**	**29**	26	**29**	**29**	24	23	23	21	24	24	24	25	**26**	19
TPP(30)	9	**13**	**13**	**13**	**13**	**13**	**13**	**13**	7	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	6	6	6	6	6	6	**8**	7	7	7
Transport(59)	36	**39**	37	36	36	32	36	35	**26**	**26**	**26**	**26**	**26**	**26**	25	**26**	**26**	24	23	25	25	24	**31**	22	24	25	25
Trucks(30)	15	16	**17**	15	15	15	15	15	9	**10**	**10**	9	9	9	9	9	9	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**
Visitall(13)	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Woodworking(20)	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	10	10	**11**	**11**	10	10	10	**11**	**11**	**11**
Zenotravel(20)	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	11	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	9	9	9	9	9	9	9	9	9	**10**
sum(1443)	1082	1110	1106	1107	1101	1101	1083	**1119**	1116	1102	825	851	847	846	879	**880**	848	854	859	837	685	699	690	695	701	715	672	711	**719**	697

Table 2: Overall coverage for GSBNB approaches on OSP IPC benchmarks. Cost bounds set to x times the best known plan cost. Best result per cost bound highlighted in **bold**.

$x = 0.25$

		GLS		GSBnB			
		SysW	SysS	$blind$	$h^{\max}$	h^{pot}	h^{car}
GLS	SysW	-	19	22	26	24	26
	SysS	0	-	17	21	19	21
GSBnB	$blind$	11	14	-	12	9	11
	$h^{\max}$	9	12	2	-	3	8
	h^{pot}	9	12	0	8	-	7
	h^{car}	9	12	1	4	4	-
best	among	16	24	24	29	28	32
	single	0	6	0	2	1	4

$x = 0.5$

		GLS		GSBnB			
		SysW	SysS	$blind$	$h^{\max}$	h^{pot}	h^{car}
GLS	SysW	-	7	18	19	19	23
	SysS	4	-	17	18	18	22
GSBnB	$blind$	17	17	-	14	10	15
	$h^{\max}$	12	13	4	-	7	12
	h^{pot}	13	14	0	8	-	8
	h^{car}	10	10	0	4	2	-
best	among	23	22	21	24	28	33
	single	2	0	0	2	1	5

$x = 0.75$

		GLS		GSBnB			
		SysW	SysS	$blind$	$h^{\max}$	h^{pot}	h^{car}
GLS	SysW	-	3	17	20	18	21
	SysS	9	-	18	21	19	22
GSBnB	$blind$	21	22	-	13	10	20
	$h^{\max}$	19	19	6	-	8	13
	h^{pot}	18	20	2	10	-	16
	h^{car}	17	18	0	2	4	-
best	among	25	21	14	19	17	25
	single	3	1	0	1	2	4

Table 3: Comparison of in how many domains of 45 each algorithm solves more instances. Cost bounds set to x times the best known plan cost. For GSBnB, the best performing configuration overall cost bound is used: GSBnB($h^{\max}, blind$), GSBnB(h^{pot}), GSBnB(h^{car}). Value v in row a_r and column a_c, means that algorithm a_c solved in v domains more instances than algorithm a_r. The last two rows depict in how many domains each algorithm is *among* the best performing algorithms and the *single* best performing algorithm respectively.

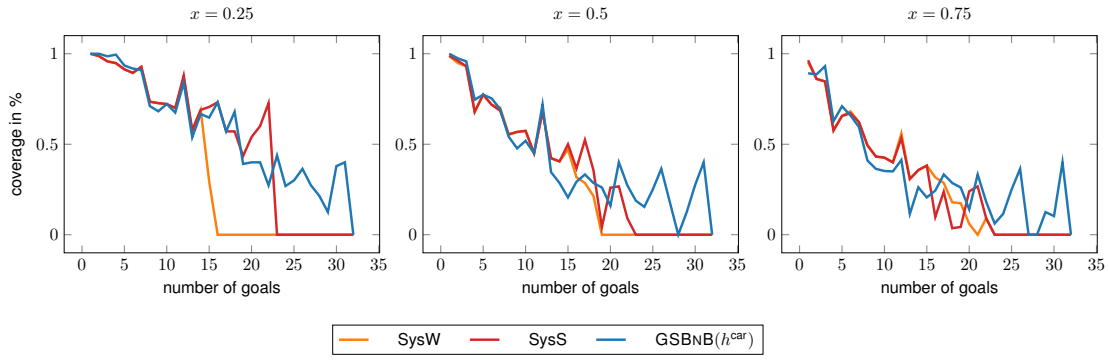


Figure 10: Relative coverage of IPC benchmarks over the number of goals for GLS with SysS and SysW and GSBnB(h^{car}). Cost bounds set to x times the best known plan cost.

For all cost bounds, both GLS expansion directions eventually drop to 0. However, for 0.5 and 0.75, there is a range of around 15 goal facts, where GLS performs better than GSBnB. For fewer goals, the results are very similar; for more goals, GSBnB solves more instances.

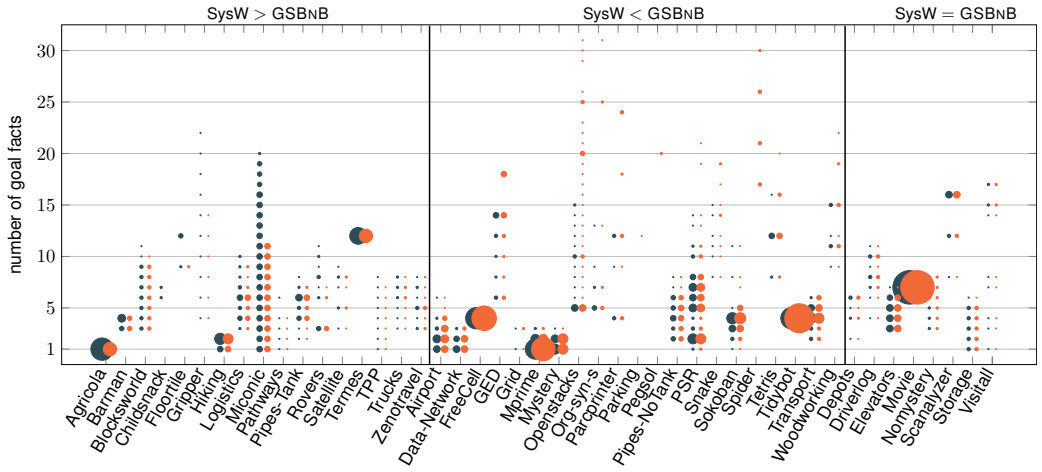


Figure 11: Comparison of number of goal facts of tasks solved by SysW(Sym) (left/blue) and GSBnB(h^{car}) (right/orange) for a cost bound of 0.75. The larger the dot the more instances with this number of goal facts have been solved. (left) domains where SysW(Sym) solved more instances than GSBnB(h^{car}); (middle) domain where GSBnB(h^{car}) solved more instances than SysW(Sym); (right) domains where both algorithms solved the same number of instances.

To investigate this further, Figure 11 depicts for a cost bound of 0.75 for SysW and GSBnB(h^{car}) the number of goal facts for a task if the approach was able to solve it. For SysW the increase in solved tasks is mainly due to Gripper and Miconic. Other domains, where SysW solved more tasks than GSBnB(h^{car}), have less than 15 soft goals. Instances that where solved by GSBnB(h^{car}) but not by SysW(Sym) have mostly more than 15 soft goals. This suggests that up to about 15 goal facts, GLS should be used, and above GSBnB.

PERFORMANCE ANALYSIS

In the following sections, we analyze the reasons behind the performance of the approaches as described in the previous sections. We begin by focusing on GLS. We compare the fraction of the goal lattice that is explored by systematic weakening SysW, which starts with all soft goals and iteratively removes them, and systematic strengthening SysS, which starts with singleton sets and iteratively adds goals). An overview of the fraction of the explored goal lattice is given in Figure 12.

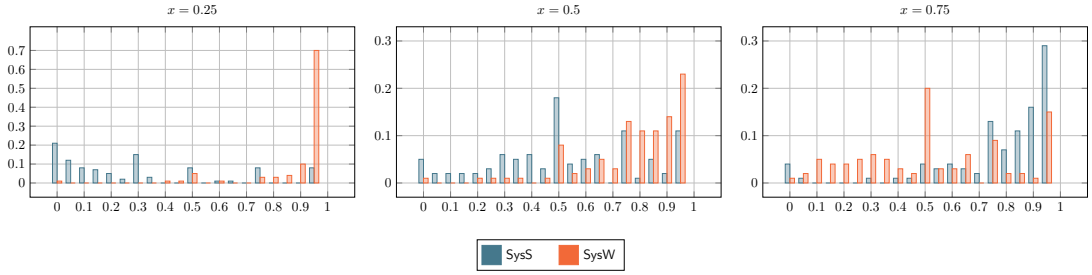


Figure 12: Comparison of the fraction of explored goal lattice. Histogram for each cost bound, bins are half open, $[x, x + 0.05)$. Cost bound x times best known solution cost.

When using SysS and a tight cost bound, only a small part of the goal lattice needs to be explored. As the cost bound increases, the fraction that needs to be explored increases as well. For SysW it is the other way around. For 0.25, SysW explores for most instances more than 95% of the goal lattice while SysS explores on average not more than a third. For 0.5 SysW still has to explore more than SysS, but for 0.75 it is less, although the difference is less prominent than for 0.25. The actual values vary significantly between the individual domains, however, the variance is typically less than 10%, suggesting that in most domains, instances exhibit similar behavior with respect to the cost bounds. For most domains, increasing the cost bound to 0.75 leads to SysW exploring a smaller fraction than SysS. The evaluation of the goal lattice fraction for each domain can be found in the Appendix A.1.

GLS with the symbolic approach, is a two-step process. First the BDD representing the reachable state space is constructed. Then, the goal lattice is traversed to identify the MUGS. The first step is independent of the expansion direction of the goal lattice. However, the generation time of the BDD representing the reachable search space, depends on the search space size and consequently on the cost bound. As shown in the top of Figure 13, the increase in time is clearly visible. The performance of the solvability check, i. e. the intersection computation, remains unaffected by its outcome. As depicted in the bottom of Figure 13 the time differences between SysS and SysW for GLS align with the difference sizes of the explored goal lattice. On average, more time is spent on the BDD computation. The difference in the explored goal lattice fractions and the time increase to generate the BDD explains the smaller difference in coverage between SysS and SysW for cost bound 0.75 compared to cost bound of 0.25. SysS solves 127 instances more with $x = 0.25$, while SysW only solves 7 instances more for $x = 0.75$.

In the following analysis we compare the average run time per goal lattice node. For the explicit search approach this is the average time to solve a task or proof unsolvability. For the symbolic approach this includes the construction of the BDD_r representing the reachable

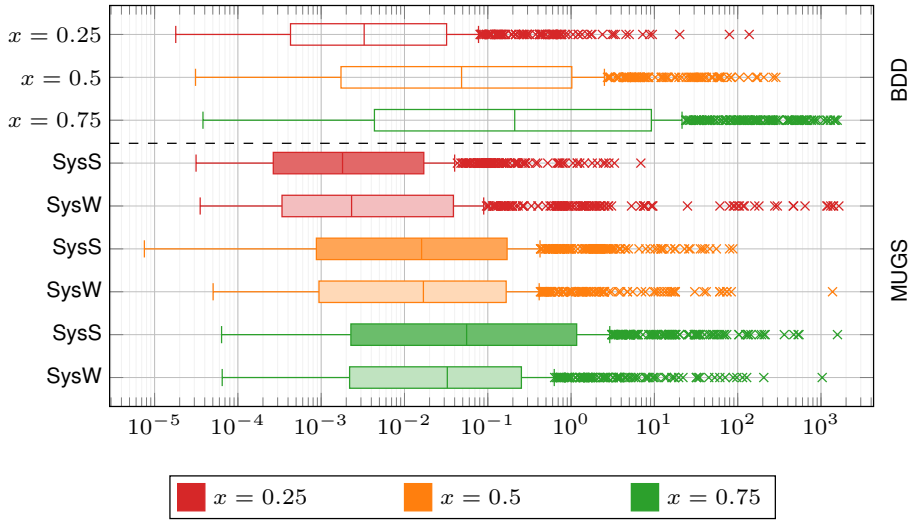


Figure 13: Runtime analysis of time to compute reachable BDD and time to perform the GLS to compute the MUGS. Only instances commonly solved by SysS and SysW over all cost bounds are considered. Cost bounds set to x times the best known plan cost, encoded by color. Top (BDD): time to compute BDD representing the reachable state space; Bottom (MUGS): time to traverse the goal lattice du compute the MUGS.

search space and then the test whether the intersection between BDD_r and the BDD_G for each soft goal subset is empty. The per-instance comparison is depicted in Figure 14. With $GBFS(h^{\text{FF}})$ finding a plan is more efficient than proving unsolvability. As seen in the top left of Figure 14, the time per goal lattice node for SysS is up to one magnitude faster than for SysW. For all approaches, the number of solved instances decreases when the cost bound increases. This is expected for SysS, but not necessarily for SysW. We would expect a smaller number of meta-search nodes and thus fewer solvability checks to lead to more solved instances; however, the search space size for each solvability check is larger for larger cost bounds and the lower number of solvability checks cannot compensate for the longer time needed per check. A comparison of the symbolic and explicit search methods (see bottom of Figure 13) demonstrated the advantage of the symbolic approach. While explicit search can be up to two orders of magnitude faster in certain instances, the symbolic approach can be up to three orders of magnitude faster.

Next we analyze the performance of the different GSBNB configurations in more detail. First we evaluate the impact of the guidance heuristic. In Figure 15 we compare the explored state space size and the search time per state for the different guidance options, i. e. no guidance, a shared heuristic for guidance and pruning, and h^{FF} for guidance. Using a guidance heuristic has clearly a positive effect on the search space size. Overall using h^{FF} for guidance decreases the search space by up to one order of magnitude. When using the same heuristic for pruning and guidance, h^{max} and h^{car} perform slightly better than h^{pot}. Using h^{FF} for guidance instead of the pruning heuristic has hardly any impact on the number of explored states (see right column of Figure 15). However, computing h^{FF} in addition to the pruning heuristic can lead to a significant increase in the time required per state, up to an order of magnitude more using h^{max} and up to two when using h^{pot} and h^{car}.

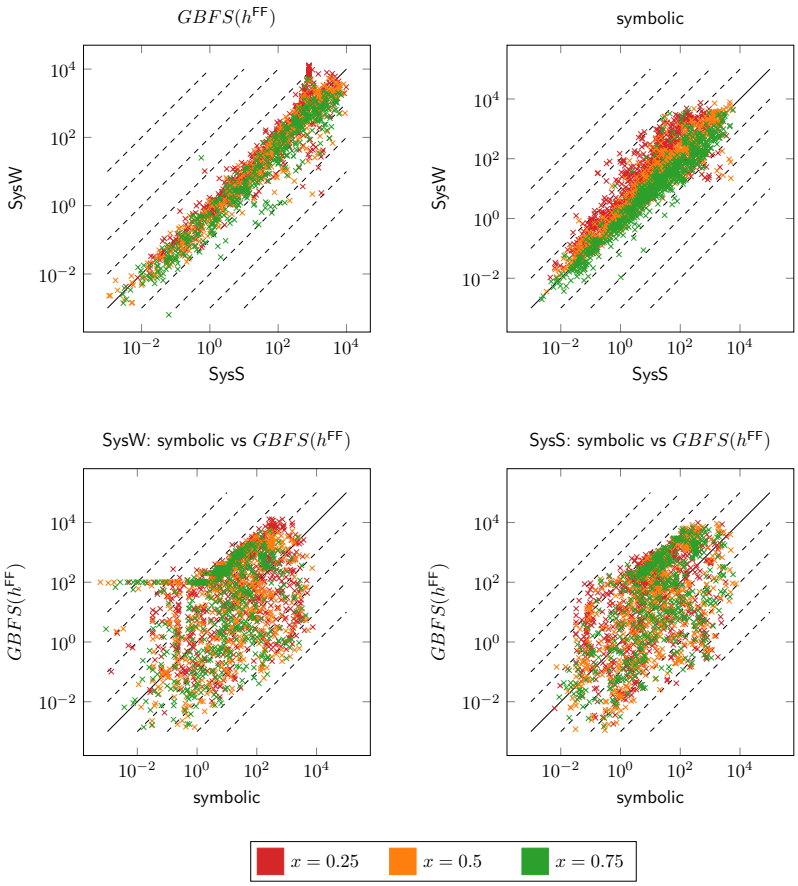


Figure 14: Comparison of time per solvability check of goal lattice node. Cost bounds set to x times the best known plan cost, encoded by color. Top: comparison between SysS and SysW; Bottom: comparison between $GBFS(h^{\mathsf{FF}})$ and Sym.

To evaluate the impact of the pruning, we compare in Figure 16 the search space sizes and search time per state for GSBNB with h^{max}, h^{pot} and h^{car} used for pruning and no guidance to GSBNB with no pruning and no guidance. All heuristics effectively reduce the search space size compared to the blind heuristic. h^{pot} performs best, followed by h^{car}, and then h^{max}. Using h^{max},h^{pot} and h^{car} reduces the search space size up to 2, 1, and 1 orders of magnitude, respectively, while on the other hand increasing the search time per state by up to 3, 1 and 1 orders of magnitude, respectively. Comparing the heuristics between each other, reveals that h^{max} and h^{car} are similarly informed, while h^{car} is much faster to compute. h^{pot} is even faster to compute, but is less informed overall.

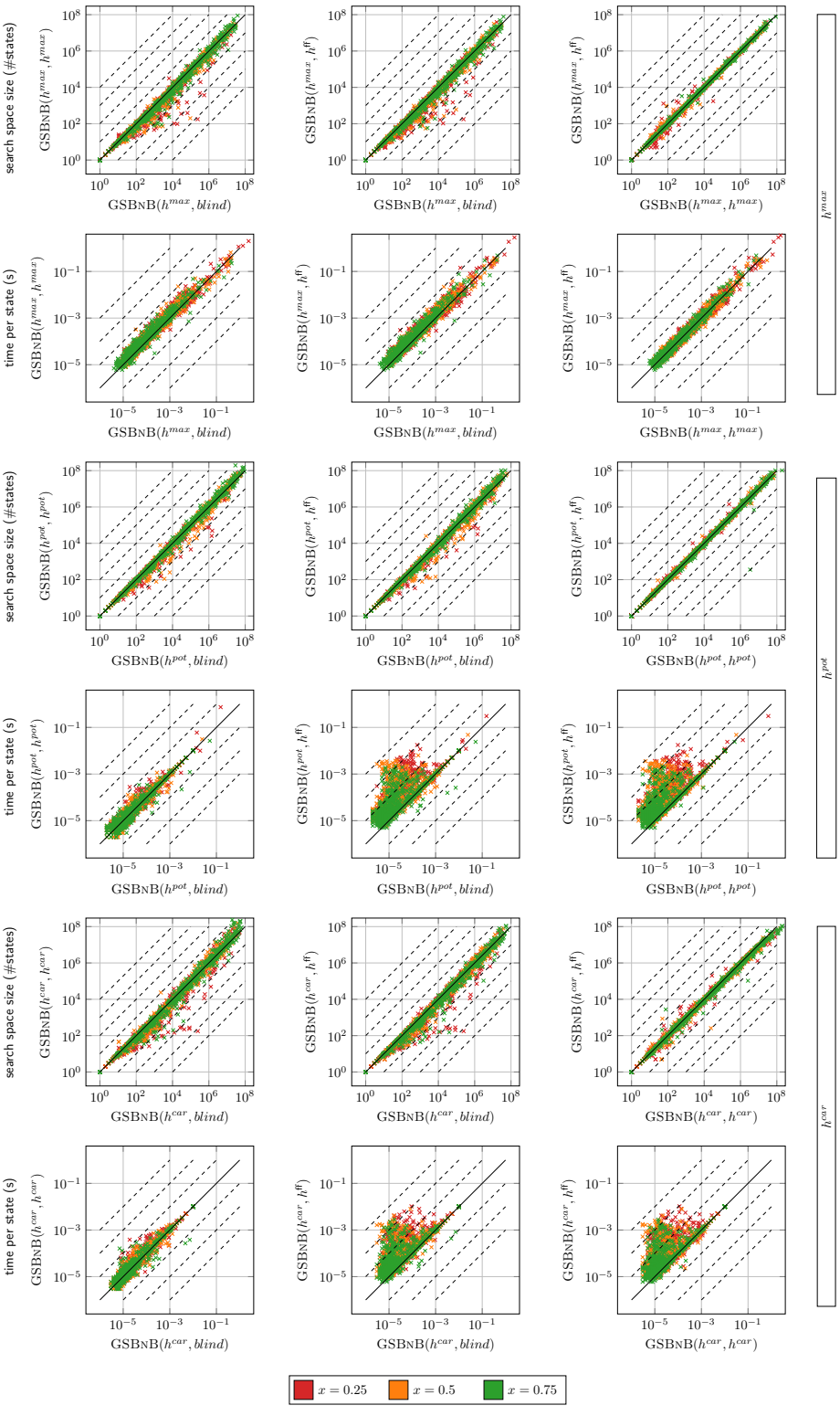


Figure 15: Per-instance comparison of number of expanded states and search time per state for GSBnB with $h^{\max}$, h^{pot} and h^{car} as pruning heuristic and either no guidance (blind) or same guidance and pruning heuristic or h^{FF} as guidance. GSBnB(h_p, h_g) where h_p is used for pruning and h_g for guidance.

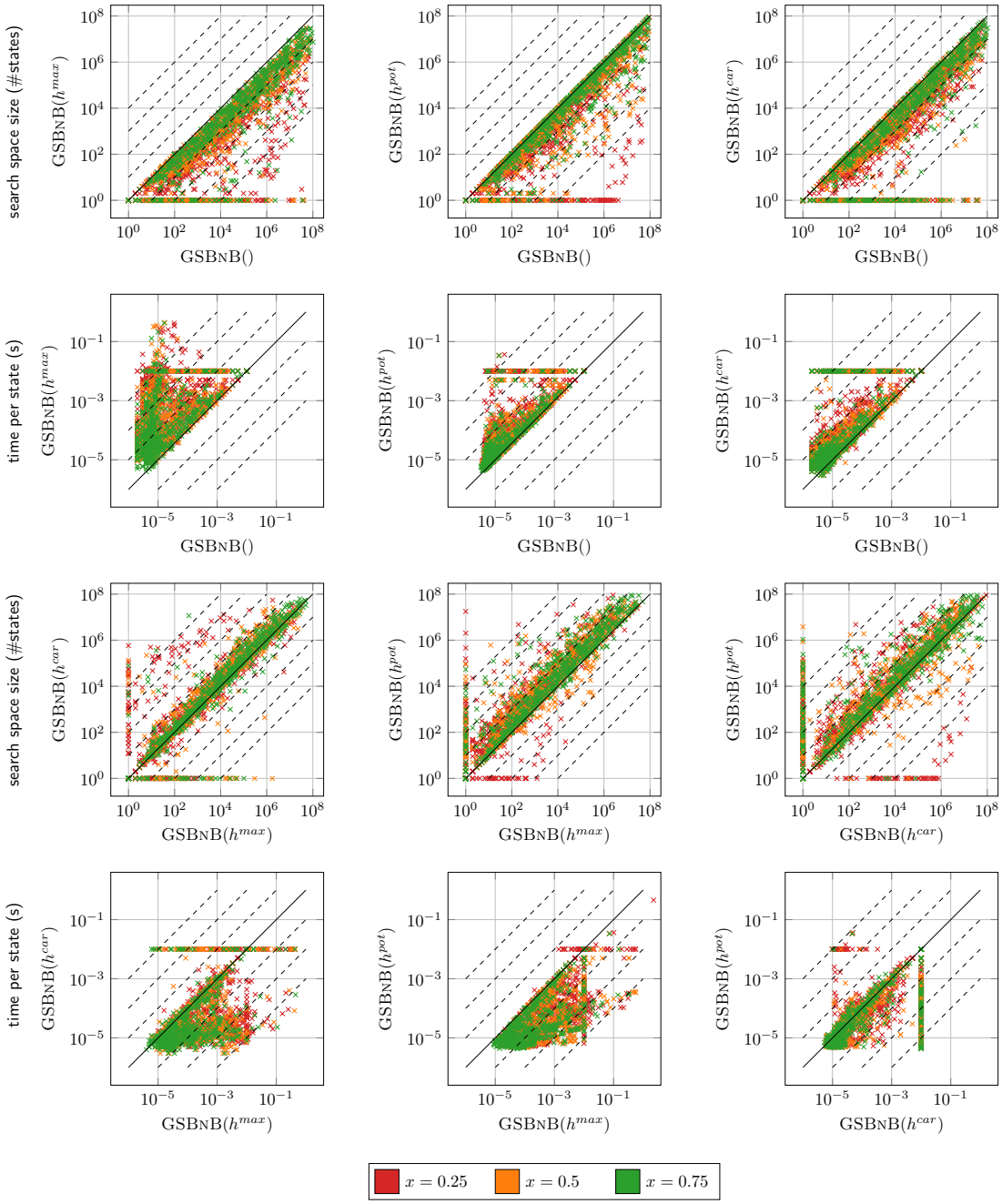


Figure 16: Per instance comparison of number of expanded states and search time per state for GSBnBwith $h^{\max}$, h^{pot} and h^{car} as pruning heuristic (GSBnB(h)) and no guidance to GSBnB with no guidance and no pruning (GSBnB()).

COMPARISON TO OSP PLANNING

By computing all maximal solvable goal subsets (MSGS), we also compute a plan for a largest solvable soft goal subset. We evaluate how our best-performing approaches compare to the state-of-the-art OSP planner. For this comparison, we use the same benchmark as before and assign a utility of *one* to each goal fact. As cost bound, we use $b = x * c$ where $x \in \{0.25, 0.5, 0.75\}$ and c is the best-known solution cost. Our version is publicly available[6]. This differs from the benchmark set used by [Katz et al., 2019], in that they introduce additional goal facts not present in the original IPC instances.

We are comparing to two different approaches for OSP planning, one based on symbolic search and one based on an explicit branch-and-bound search. The approach by Katz et al. [2019] (BnB[7]) uses a branch-and-bound search on a reformulation of the OSP task with multiple cost functions, but no utility function. They use two heuristics, one for guidance and one for pruning. We use the best-performing configuration using the blind heuristic for both. The state-of-the-art OSP planner (Sym[8]) by Speck and Katz [2021], is similar to our symbolic approach, performing an exhaustive symbolic forward search. However, since they only try to find one goal subset with the highest utility, or in the spacial case here, the largest goal subset, they keep track of it during the state space exploration instead of the extraction during GLS as in our case.

We use our most effective approaches based on GLS using symbolic search (SysS and SysW), and GSBNB, with the same heuristic for guidance and pruning, h^{pot} and h^{car}. To provide an optimal plan to a MSGS with maximal size, the following modifications are made. For GLS, during the goal lattice exploration a maximally sized MSGS is collected and afterwards a plan to it is extracted as done in symbolic search for a classical task. In GSBNB, we maintain a currently maximally sized MSGS and the state in which it is satisfied during the states space exploration. After the search, the optimal plan leading to that state is reconstructed.

Table 4 contains the coverage results. An instance counts as solved if an optimal plan for a largest soft goal subset could be computed within the time and memory limit. BnB is for all cost bounds the worst performing approach. Sym performs better for a cost bound of 0.75, solving 29 instances more than our best approach GSBNB(h^{car}). For the smaller cost bounds we outperform Sym by solving 57 and 54 additionally instances for cost bounds 0.25 and 0.5 respectively.

As shown in Table 5, we examine the number of domains where each approach performs best. It shows that our MUGS computation approaches, especially GSBNB, are orthogonal to Sym. For the smaller cost bounds of 0.25 and 0.5, GSBNB performs better in significantly more domains than Sym. For the largest cost bound of 0.75, Sym and GSBNB(h^{car}) are orthogonal, performing better in 21 and 18 tasks respectively.

This indicates, that computing allMUGS has in practice a similar difficulty to solving the OSP task itself.

[6] https://doi.org/10.5281/zenodo.14988342
[7] https://doi.org/10.5281/zenodo.3359211
[8] https://github.com/speckdavid/symbolic-osp

	$x = 0.25$							$x = 0.5$							$x = 0.75$						
	OSP		MUGS			GSBnB		OSP	MUGS			GSBnB			OSP	MUGS			GSBnB		
	Sym	BnB	GLS			h^{pot}	h^{car}	Sym	BnB	GLS		h^{pot}	h^{car}		Sym	BnB	GLS			h^{pot}	h^{car}
domain			SysW	SysS						SysW	SysS						SysW	SysS			
Agricola(20)	6	0	**20**	**20**		**20**	**20**	6	0	**20**	**20**	**20**	**20**		6	0	**20**	**20**		17	12
Airport(50)	22	27	23	23		29	**33**	20	24	21	21	**27**	**27**		20	21	19	18		24	**25**
Barman(34)	15	10	24	24		18	**25**	11	7	**13**	**13**	11	11		**11**	4	**11**	**11**		4	8
Blocksworld(35)	**35**	**35**	32	**35**		**35**	**35**	28	27	**29**	**29**	**29**	**29**		21	21	**23**	**23**		21	22
Childsnack(20)	**6**	0	**6**	2		2	2	**4**	0	**4**	0	0	0		**4**	0	**4**	**4**		0	0
Data-Network(20)	14	4	19	19		19	**20**	14	1	**17**	**17**	**17**	**17**		14	1	13	13		13	**15**
Depots(22)	14	16	15	15		17	**18**	7	11	11	9	**12**	**12**		5	**7**	**7**	**7**		**7**	**7**
Driverlog(18)	**15**	**15**	**15**	**15**		**15**	**15**	**14**	13	**14**	**14**	**14**	**14**		11	10	**12**	**12**		11	**12**
Elevators(30)	21	18	**30**	**30**		**30**	**30**	14	10	**25**	**25**	**25**	**25**		14	10	**24**	**24**		**24**	**24**
Floortile(36)	9	7	11	**18**		16	16	2	2	**11**	9	6	6		2	2	**6**	4		2	2
FreeCell(80)	65	76	76	76		**80**	**80**	24	29	29	29	**71**	36		15	20	19	18		**31**	22
GED(20)	**20**	**20**	15	**20**		**20**	**20**	15	19	19	19	**20**	**20**		**20**	**20**	15	15		**20**	**20**
Grid(5)	**5**	**5**	**5**	**5**		**5**	**5**	3	3	3	3	3	**4**		2	2	2	2		2	**3**
Gripper(14)	**14**	11	6	10		7	7	**14**	8	8	8	6	6		**14**	8	10	7		6	6
Hiking(20)	19	18	19	19		**20**	**20**	16	14	16	17	16	**18**		**15**	12	**15**	**15**		13	14
Logistics(61)	34	27	37	**38**		33	36	**26**	20	**26**	**26**	21	24		21	14	**23**	**23**		18	19
Miconic(150)	**132**	94	75	110		93	93	**101**	64	90	89	64	65		**95**	55	94	77		55	55
Movie(30)	**30**	**30**	**30**	**30**		**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**		**30**	**30**	**30**	**30**		**30**	**30**
Mprime(35)	**35**	**35**	**35**	**35**		**35**	**35**	31	28	34	34	33	**35**		27	24	28	29		28	**31**
Mystery(19)	**19**	**19**	**19**	**19**		**19**	**19**	18	**19**	**19**	**19**	**19**	**19**		15	17	17	17		18	**19**
Nomystery(20)	**20**	**20**	**20**	**20**		**20**	**20**	16	14	16	16	14	**18**		**12**	9	**12**	**12**		10	**12**
Openstacks(58)	**58**	44	17	38		45	50	**55**	42	17	38	40	40		**54**	42	19	38		36	36
Org-syn-s(15)	10	9	8	8		**13**	10	**10**	9	8	8	**10**	9		**10**	9	8	8		9	9
Parcprinter(15)	**15**	9	7	9		**15**	**15**	**15**	9	7	7	13	14		**15**	9	7	7		12	12
Parking(40)	16	25	14	22		**26**	**26**	0	2	2	2	**7**	**7**		0	0	0	0		**1**	**1**
Pathways(24)	**7**	5	**7**	**7**		5	6	**5**	4	**5**	**5**	4	**5**		**5**	4	**5**	**5**		4	4
Pegsol(2)	**2**	**2**	0	1		**2**	**2**	**2**	**2**	0	1	**2**	**2**		**2**	**2**	0	0		**2**	**2**
Pipes-NoTank(50)	40	45	45	45		**46**	**46**	23	30	29	29	**36**	**36**		16	21	22	22		**24**	**24**
Pipes-Tank(50)	37	33	**41**	**41**		40	38	20	18	**24**	**24**	**24**	**24**		17	16	**18**	**18**		16	16
PSR(50)	**50**	**50**	48	48		**50**	**50**	**50**	**50**	48	48	**50**	**50**		**50**	**50**	48	48		**50**	**50**
Rovers(31)	**19**	15	**19**	**19**		16	16	**14**	8	**14**	**14**	9	9		**14**	6	**14**	**14**		7	7
Satellite(19)	**16**	9	12	15		11	11	11	7	**12**	**12**	7	7		**7**	6	**7**	**7**		6	6
Scanalyzer(22)	10	10	4	11		**14**	13	**10**	**10**	**10**	**10**	**10**	**10**		**10**	**10**	**10**	4		**10**	**10**
Snake(17)	**17**	**17**	6	8		**17**	**17**	10	**17**	8	7	15	15		7	**14**	6	6		10	10
Sokoban(30)	**30**	**30**	**30**	**30**		**30**	**30**	**30**	**30**	27	27	**30**	**30**		**30**	**30**	23	23		26	28
Spider(12)	**12**	**12**	0	6		**12**	**12**	**12**	**12**	0	6	**12**	**12**		**12**	**12**	0	6		11	11
Storage(30)	18	20	19	20		**22**	**22**	16	17	17	17	**18**	**18**		15	15	**16**	**16**		**16**	**16**
Termes(20)	**20**	19	**20**	**20**		**20**	**20**	17	15	**18**	**18**	16	16		**15**	12	**15**	**15**		11	12
Tetris(17)	10	**16**	7	15		**16**	**16**	6	10	8	9	**13**	**13**		5	7	8	7		10	**11**
Tidybot(30)	**30**	**30**	**30**	**30**		**30**	**30**	22	**29**	27	27	**29**	**29**		12	24	19	19		24	**26**
TPP(30)	12	9	12	12		**13**	**13**	**8**	7	**8**	**8**	**8**	**8**		**8**	6	**8**	**8**		**8**	7
Transport(59)	19	13	**36**	**36**		**36**	35	18	13	**26**	**26**	**26**	**26**		19	13	23	23		**31**	25
Trucks(30)	15	14	17	**19**		15	15	10	8	**12**	**12**	9	9		**9**	6	**9**	**9**		8	8
Visitall(13)	**13**	**13**	6	10		**13**	**13**	**13**	**13**	8	10	10	10		**13**	12	10	9		10	10
Woodworking(20)	**20**	7	5	12		18	18	**20**	7	8	8	12	12		**20**	7	8	7		10	11
Zenotravel(20)	13	13	**14**	**14**		13	13	**12**	10	**12**	**12**	**12**	**12**		9	8	**10**	**10**		9	9
sum(1443)	1059	956	956	1083		1101	**1116**	828	723	806	836	**880**	859		**748**	628	717	710		715	719

Table 4: Coverage comparison of cardinal maximal OSP planning to allMUGS computation on IPC benchmarks modified with a plan-cost bound. Cost bounds set to x times the best known plan cost. Best performance within each cost bound shown in **boldface**.

$x = 0.25$

			OSP		MUGS			
			Sym	BnB	GLS		GSBnB	
					SysW	SysS	h^{pot}	h^{car}
OSP		Sym	-	7	15	18	18	18
OSP		BnB	22	-	17	20	26	27
MUGS	GLS	SysW	17	19	-	19	24	26
MUGS	GLS	SysS	12	12	0	-	19	21
MUGS	BnB	h^{pot}	9	2	9	12	-	7
MUGS	BnB	h^{car}	8	2	9	12	4	-
best		among	25	16	17	22	29	30
best		single	5	0	0	3	2	4

$x = 0.5$

			OSP		MUGS			
			Sym	BnB	GLS		GSBnB	
					SysW	SysS	h^{pot}	h^{car}
OSP		Sym	-	10	21	22	18	21
OSP		BnB	26	-	23	23	27	30
MUGS	GLS	SysW	13	14	-	7	19	23
MUGS	GLS	SysS	13	15	4	-	18	22
MUGS	BnB	h^{pot}	14	4	13	14	-	8
MUGS	BnB	h^{car}	13	4	10	10	2	-
best		among	21	11	21	20	26	29
best		single	5	1	1	0	1	4

$x = 0.75$

			OSP		MUGS			
			Sym	BnB	GLS		GSBnB	
					SysW	SysS	h^{pot}	h^{car}
OSP		Sym	-	8	17	17	14	18
OSP		BnB	26	-	27	24	22	26
MUGS	GLS	SysW	15	13	-	3	18	21
MUGS	GLS	SysS	16	15	9	-	19	22
MUGS	BnB	h^{pot}	21	7	18	20	-	16
MUGS	BnB	h^{car}	20	6	17	18	4	-
best		among	24	9	22	20	13	19
best		single	7	1	1	0	2	7

Table 5: Comparison of in how many domains out of 45 each algorithm solves more instances, from top to bottom for cost bounds 0.25, 0.5 and 0.75. Value v in row a_r and column a_c, means than algorithm a_c solved in v domains more instances than algorithm a_r. The last two rows depict in how many domains each algorithm is *among* the best performing algorithms and the *single* best performing algorithm respectively.

4.3.2 TEMPORAL GOALS

In the conflict analysis, we have previously focused on soft goals, represented by individual facts, such as `image-ice` = `uploaded`. However, often this is not sufficient to fully express a user's preferences. For example, it does not capture preferences like the `image-ice` to be uploaded before `soil-sample-ice` or that the rover never visiting the crater. In the following, we discuss two languages for defining such temporal goals and how they can be compiled into the planning task, to make them accessible to the conflict analysis.

COMPILING TEMPORAL GOALS INTO GOAL FACTS

To include temporal goals in the conflict analysis, it is necessary to indicate their satisfaction by a single fact g_{p_t}. These facts can then be incorporated as soft goals in G^{soft}, and the corresponding MUGS can be computed with one of the introduced algorithms. To apply this procedure, a temporal goal compilation must satisfy the following constraints.

DEFINITION 25: TEMPORAL GOAL COMPILATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and p_t a temporal goal. Then the task $\tau_{p_t} = (V_{p_t}, A_{p_t}, c_{p_t}, I_{p_t}, G^{\text{hard}}_{p_t}, G^{\text{soft}}_{p_t}, b_{p_t})$ with $g_{p_t} \in G^{\text{soft}}_{p_t}$ is a **temporal goal compilation** of p_t with plan mapping $\beta : A^*_{p_t} \mapsto A^*$ where X^* is any sequence of elements from X, if

(1) $\forall \pi \in \Pi(\tau) : \pi \vDash p_t \rightarrow \exists \pi' \in \Pi(\tau_{p_t}) : \beta(\pi) = \pi' \wedge g_{p_t} \in I_{p_t}[\![\pi']\!]$.

(2) $\Pi(\tau) = \{\beta(\pi') \mid \pi' \in \Pi(\tau_{p_t})\}$,

(3) $\forall \pi \in \Pi(\tau_{p_t}) : G^{\text{soft}} \cap I_{p_t}[\![\pi]\!] = G^{\text{soft}} \cap I[\![\beta(\pi)]\!]$

Condition (1) ensures that for all plans that satisfy the temporal goal p_t, there is a corresponding plan in the compilation that satisfies the fact g_{p_t}. The compilation must preserve the plans (2) and the achieved soft goals (3). Otherwise, the compilation would compromise the satisfiability of soft goal subsets and introduce new conflicts that are not necessarily caused by the unsatisfiability of p_t.

PROPOSITION 9: TEMPORAL GOAL COMPILATION: MUGS CORRECTNESS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and p_t a temporal goal and task $\tau_{p_t} = (V_{p_t}, A_{p_t}, c_{p_t}, I_{p_t}, G^{\text{hard}}_{p_t}, G^{\text{soft}}_{p_t}, b_{p_t})$ with $g_{p_t} \in G^{\text{soft}}_{p_t}$ a temporal goal compilation of p_t. Then

$$\mathcal{G}^{\text{MUGS}}(\tau_{p_t}) = \mathcal{G}^{\text{MUGS}}(\tau) \cup \{G \cup g_{p_t} \mid G \subseteq G^{\text{soft}} \wedge \text{IS-MUGS}(G, p_t)\}$$

where $\text{IS-MUGS}(G, p) = (\forall \pi \in \Pi(\tau) : G \subseteq I[\![\pi]\!] \rightarrow \pi \nvDash p) \wedge (\forall G' \subsetneq G : \exists \pi \in \Pi(\tau) : G' \subseteq I[\![\pi]\!] \wedge \pi \vDash p)$

Proof:
Let (1), (2) and (3) be the requirements from Definition 25.

$\supseteq$: Let $C \in \mathcal{G}^{\text{MUGS}}(\tau)$. C is unsolvable and minimal in τ_{p_t}, because based on (2) and (3) all soft goal subsets $G \subseteq G^{\text{soft}}$ that are solvable/unsolvable in τ are also solvable/unsolvable in τ_{p_t}.

Let $C \in \{G \cup g_{p_t} \mid G \subseteq G^{\text{soft}} \wedge \text{IS-MUGS}(G, p_t)\}$. C is unsolvable in τ_{p_t}. All plans $\pi \in \Pi(\tau)$ that satisfy $C \setminus \{g_{p_t}\}$ do not satisfy p_t and from (2) follows for all plans $\pi \in \Pi(\tau_{p_t}) : \beta(\pi) \nvDash p_t \rightarrow g_{p_t} \notin I_{g_{p_t}}[\![\pi]\!]$. Thus, based on (2) and (3) there is no $\pi \in \Pi(\tau_{p_t})$ such that $C \subseteq I_{g_{p_t}}[\![\pi]\!]$. C is minimal because for all $G' \subsetneq C$ there exists a plan $\pi \in \Pi(\tau_{p_t})$ such that $G' \cup \{g_{p_t}\} \subseteq I_{p_t}[\![\pi]\!]$. This holds, because there exists a $\pi \in \Pi(\tau)$ such that $G' \subseteq I[\![\pi]\!] \wedge \pi \vDash p_t$. From (1) follows there exists a plan $\pi' \in \Pi(\tau_{p_t})$ such that $g_{p_t} \in I_{p_t}[\![\pi']\!]$ and because of (2) and (3) $G' \subseteq I_{p_t}[\![\pi']\!]$.

$\subseteq$: Let $C \in \mathcal{G}^{\text{MUGS}}(\tau_{p_t})$. This means there is no plan $\pi \in \Pi(\tau_{p_t})$ such that, $C \subseteq I_{p_t}[\![\pi]\!]$.

$g_{p_t} \notin C$: C is unsolvable and minimal in τ because of (2) and (3) in and thus $C \in \mathcal{G}^{\text{MUGS}}(\tau)$.

$g_{p_t} \in C$: For all plans $\pi' \in \Pi(\tau_{p_t})$ where $C \setminus \{g_{p_t}\} \subseteq I_{p_t}[\![\pi']\!]$ it holds $g_{p_t} \notin I_{p_t}[\![\pi']\!]$. From (2) and (3) in follows that $C \setminus \{g_{p_t}\} \subseteq I[\![\beta(\pi')]\!]$ and from (1) that $\beta(\pi') \nvDash p_t$. (Otherwise there would be a plan in $\Pi(\tau_{p_t})$ that satisfies C.) For all $G \subsetneq C$, $G \cup \{g_{p_t}\}$ is solvable τ_{p_t} because C is minimal. Thus, from (2) and (3) in follows that there exists a plan π such that $G \subseteq I[\![\pi]\!]$ and from (1) that $\pi \vDash g_{p_t}$.

□

In the following we discuss two languages, to define temporal goals, provide compilations into soft goal facts and evaluate their performance with different allMUGS computation approaches. First, we explore LTL_f, a version of *Linear Temporal Logic* (LTL) [Huth and Ryan, 2004] for finite traces. LTL_f allows the expression of complex temporal soft goals. Secondly, we consider a simpler language based on used action subsets. While less expressive, it can lead to a more efficient compilation.

LTL_F Soft Goals

A common language to express temporal preferences over infinite traces in model checking is *Linear Temporal Logic* (LTL) [Huth and Ryan, 2004]. In planning, however, plans represent finite traces. We follow Baier and McIlraith [2006a] and adopt their *finite* LTL (LTL_f) language. It is an adaptation of LTL with a semantic over finite traces. For syntax and semantics of LTL_f, we refer to Section 3.2. Example 11 outlines some temporal preferences for our running example, expressed in LTL_f.

EXAMPLE 11: TEMPORAL SOFT GOALS USING LTL_F

We give examples of some commonly used temporal goals, also provided as temporal preferences in PDDL3 [Gerevini et al., 2009] and possible instantiations in our running example. We assume that there are multiple rovers.

- Never: $\Box\neg a$
  - Do not use the connection between the crater and the rock: $\Box\neg(\texttt{drive}(\texttt{crater1},\texttt{rock}) \vee \texttt{drive}(\texttt{rock},\texttt{crater1}))$
  - The rovers should never be at the same location: $\Box\neg(\bigvee_{l\in D_R} \texttt{R1} = l \wedge \texttt{R2} = l)$
- Eventually: $\Diamond a$
  - Rover R1 should visit the rock: $\Diamond\texttt{R1} = \texttt{rock}$
  - Rover R1 should take the image-ice: $\Diamond\texttt{image}(\texttt{R1},\texttt{image-ice})$
- a before b: $\neg b$ U a
  - image-ice uploaded before the soil-sample-rock: $\neg(\texttt{soil-sample-rock} = \texttt{upload})$ U $(\texttt{image-ice} = \texttt{upload})$
- At most once a: $\Box(a \rightarrow (a \text{ W } \Box\neg a))$
  - Visit the rock at most once: $\Box((\texttt{R1} = \texttt{rock} \vee \texttt{R2} = \texttt{rock}) \rightarrow ((\texttt{R1} = \texttt{rock} \vee \texttt{R2} = \texttt{rock}) \text{ W } \Box\neg(a\texttt{R1} = \texttt{rock} \vee \texttt{R2} = \texttt{rock})))$

More examples for commonly used LTL_f templates can be found in Table 14 and the temporal soft goals we used in our evaluation in Table 16.

LTL_f is interpreted over states. To include actions, as in some examples in Example 11, you can add action effects that identify the last action applied.

DEFINITION 26: APPLIED ACTIONS COMPILATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and $\bar{A} \subseteq \mathcal{P}(A)$ a set of action subsets. Then $\tau' = (V', A', c, I', G^{\text{hard}}, G^{\text{soft}}, b)$ with $V_{\bar{A}} = \{v_A \mid A \in \bar{A}\}$ and $D(v_A) = \{\top, \bot\}$ for every $A \in \bar{A}$

- $V' = V \cup V_{\bar{A}}$
- $I' = I \cup \{v_A = \bot \mid v_A \in V_{\bar{A}}\}$
- $A' = \{a' \mid a \in A\}$
  - $pre_{a'} = pre_a$

– $\mathit{eff}_{a'} = \mathit{eff}_a \cup \{v_A = \top | v_A \in V_{\bar{A}}, a \in A\} \cup \{v_A = \bot | v_A \in V_{\bar{A}}, a \notin A\}$

is the **applied actions compilation** for action subsets $\bar{A}$.

A common approach to compile LTL$_f$ soft goal p_t into a planning task τ with actions A has two steps:

(1) Produce an automaton accepting traces that satisfy the LTL$_f$ soft goal p_t.

(2) Compile the automaton into additional variables and actions A_{p_t} and enforce an alternating execution of actions in A_{p_t} and A.

For step (1) we use Baier and McIlraith [2006a]'s tool. It produces a non-deterministic finite automaton (NFA). For step (2), we cannot use Baier and McIlraith [2006a] compilation as it relies on PDDL axioms [Thiebaux et al., 2005] which we do not support. Instead, we are using a compilation introduced by Edelkamp [2006], as it only adds and modifies variables and actions. In the following, we assume that the NFA guards are given in disjunctive normal form with only positive literals.

DEFINITION 27: LTL$_F$-COMPILATION BASED ON EDELKAMP [EDELKAMP, 2006]

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and G_{LTL_f} a set of LTL$_f$ temporal goals. Given $\mathcal{N}$ a list of NFAs $N^i = (S^i, \Sigma, T^i, s_0^i, S_a^i)$ accepting traces satisfying LTL$_f$ soft goals $p_t^i \in G_{LTL_f}$, then $\tau' = (V', A', c', I', G^{\text{hard}\prime}, G^{\text{soft}\prime}, b)$ is the LTL$_f$-compilation where

- $V' = V \cup \bigcup_{N \in \mathcal{N}} \{v_{S^N}, v_{a^N}\} \cup \{\mathit{sync}\}$ where $D(v_{S^{N^i}}) = S^i$, $D(v_{a^N}) = \{\top, \bot\}$ and $D(\mathit{sync}) = \mathcal{N} \cup \{\tau\}$
- $A' = A_\tau \cup \bigcup_{N^i \in \mathcal{N}\rangle} A_N$
  - $A_\tau = \{a_\tau \mid a \in A\}$
    - $\mathit{pre}_{a_\tau} = \mathit{pre}_a \cup \{\mathit{sync} = \tau\}$
    - $\mathit{eff}_{a_\tau} = \mathit{eff}_a \cup \{\mathit{sync} = N^0\}$
  - $A_{N^i} = \{a_t^c \mid t = (s, g, s') \in T^i, c \in g\}$
    - $\mathit{pre}_{a_t^c} = \{S^{N^i} = s, \mathit{sync} = N^i\} \cup c$
    - $\mathit{eff}_{a_t^c} = \{S^{N^i} = s'\} \cup \{v_{a^i} = (s' \in S_{a^{N^i}})\} \cup \{\mathit{sync} = x \mid x = N^{i+1} \text{ if } N^{i+1} \in \mathcal{N} \text{ else } \tau\}$
- $c'(a) = 0$ for all $a \in \bigcup_{N^i \in \mathcal{N}\rangle} A_N$ and $c'(a) = c(a)$ for all $a \in A_\tau$
- $I' = I \cup \{\mathit{sync} = N^0\} \cup \{v_{s_0^{N^i}} = \top, v_{a^{N^i}} = \bot \mid N^i \in \mathcal{N}\}$
- $G^{\text{hard}\prime} = G^{\text{hard}} \cup \{\mathit{sync} = \tau\}$
- $G^{\text{soft}\prime} = G^{\text{soft}} \cup \{v_{a^{N^i}} = \top \mid N^i \in \mathcal{N}\}$

To satisfy Definition 25 it is necessary that *sync* $= \tau$ is a hard goal. Otherwise, it is not guaranteed, that all NFAs have been evaluated before $v_{a^i} = \top$ is used to check if p_t^i is satisfied.

Another important detail is that, as temporal goals are soft goals in our context, we need to continue search paths even when one of the automata cannot reach an accepting state anymore. Edelkamp [2006] achieves this by adding an action for each automaton that results in a dead state, allowing future synchronizations with that automaton to be skipped. Instead, we address this problem in step (1) by converting Baier and McIlraith [2006a] initial NFA into a *complete* NFA. This means that there is an applicable transition in each state. This is a standard operation that adds a new state with a self-loop with guard `true`, which is reached via a new transition if none of the existing transitions are applicable.

PROPOSITION 10: CORRECTNESS LTL$_F$ COMPILATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, G_{LTL_f} a set of LTL$_f$ temporal goals and $\tau' = (V', A', c', I', G^{\text{hard}'}, G^{\text{soft}'}, b)$ is the LTL$_f$-compilation based on Definition 27.
Let the plan mapping $\beta : A'^* \mapsto A^*$ be defined as $\beta(\pi = a_0 \cdots a_n) = \beta(a_0) \cdots \beta(a_n)$ with

$$\beta(a) = \begin{cases} \epsilon & \text{if } a \in A_N \\ a' \text{ with } pre'_a = pre_a[A] \text{ and } \mathit{eff}'_a = \mathit{eff}_a[A] & \text{otherwise} \end{cases}$$

where $P[V]$ is the projection of partial assignment P to the variables in V.
Task τ' is a temporal goal compilation according to Definition 25.

For the proof see Appendix A.2.1.

ACTION-SET SOFT GOALS

Two very common temporal goals are to *eventually* perform action a and to *never* perform action a, which correspond to the LTL$_f$ formulas $\Diamond a$ and $\Box\neg a$. For example, the temporal goals can be used to express that the rover should visit a certain place or that a certain connection must be avoided, see Example 11. To determine whether these temporal goals are satisfied, it is not necessary to know the sequence of events but only whether they appeared. For this special case we introduce *Action-Set* soft goals.

DEFINITION 28: ACTION-SET SOFT GOALS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, Π its set of plans, and $A_1, \ldots, A_n \subseteq A$.
An **Action-Set soft goal** for τ and $A_1, \ldots, A_n$ is a function $g_\phi : \Pi \to \{\top, \bot\}$. ϕ is a propositional formula over the atoms $A_1, \ldots, A_n$, and $g_\phi(\pi) = \top$ iff ϕ evaluates to $\top$ under the truth value assignment where A_i is $\top$ iff π contains at least one action from A_i.

Action-Set soft goals are compiled by introducing two phases: the *planning phase* and the *evaluation phase*, and an action to switch from planning to evaluation phase. One variable $isUsed_i$ per action set tracks during the planning phase which action sets have been used. In the evaluation phase actions evaluate each g_ϕ based on the $isUsed_i$ flags (following [Gazen and Knoblock, 1997, Nebel, 2000]). The result is stored in $isTrue_\phi$ which can be used as a soft goal. In the compilation, we assume that ϕ is given in disjunctive normal form.

DEFINITION 29: ACTION-SET COMPILATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task. Let G_{AS} be a set of Action-Set soft goals and $A_1, \ldots, A_n \subseteq A$ the used action subsets.
Then $\tau' = (V', A', c', I', G^{\text{hard}'}, G^{\text{soft}'}, b)$ is the compilation where

- $V' = V \cup \{\textit{phase}\} \cup \bigcup_{A_i} \{isUsed_i\} \cup \bigcup_{g_\phi \in G_{AS}} \{isTrue_\phi\}$
  where $D(isUsed_i) = D(isTrue_\phi) = \{\top, \bot\}$ and $D(phase) = \{\textit{plan}, \textit{eval}\}$
- $A' = A_\tau \cup \{a_{cp}\} \cup \bigcup_{g_\phi \in G_{AS}} A_{g_\phi}$
  - $A_\tau = \{a_\tau | a \in A\}$
    - $pre_{a_\tau} = pre_a \cup \{\textit{phase} = \textit{plan}\}$
    - $\mathit{eff}_{a_\tau} = \mathit{eff}_a$
  - a_{cp}
    - $pre_{a_{cp}} = \{\textit{phase} = \textit{plan}\}$
    - $\mathit{eff}_{a_{cp}} = \{\textit{phase} = \textit{eval}\}$
  - $A_{g_\phi} = \{a_C \mid C \in \text{clauses of } \phi\}$
    - $pre_{a_C} = \{isUsed_i = \top \mid A_i \in C\} \cup \{isUsed_i = \bot \mid \neg A_i \in C\}$
    - $\mathit{eff}_{a_t} = \{isTrue_\phi = \top\}$
- $c'(a) = 0$ for all $a \in \{a_{cp}\} \cup \bigcup_{g_\phi \in G_{AS}} A_{g_\phi}$ and $c'(a) = c(a)$ for all $a \in A_\tau$
- $I' = I \cup \{\textit{phase} = \textit{plan}\} \cup \{isUsed_i = \bot \mid A_i\} \cup$
  $\{isTrue_\phi = \bot \mid g_\phi \in G_{AS}\}$
- $G^{\text{hard}'} = G^{\text{hard}}$
- $G^{\text{soft}'} = G^{\text{soft}} \cup \{isTrue_\phi = \top \mid g_\phi \in G_{AS}\}$

PROPOSITION 11: CORRECTNESS ACTION-SET COMPILATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, G_{LTL_f} a set of LTL$_f$ temporal goals and $\tau' = (V', A', c', I', G^{\text{hard}'}, G^{\text{soft}'}, b)$ is the Action-Set compilation based on Definition 29.

The plan mapping $\beta : A'^* \mapsto A^*$ is given by $\beta(\pi = a_0 \cdots a_n) = \beta(a_0) \cdots \beta(a_n)$ with

$$\beta(a) = \begin{cases} \epsilon & \text{if } a \in A_{g_\phi} \\ a' \text{ with } pre'_a = pre_a[A] \text{ and } eff'_a = eff_a[A] & \text{otherwise} \end{cases}$$

where $P[V]$ is the projection of partial assignment P to the variables in V.
Task τ' is a temporal goal compilation according to Definition 25.

For the proof see Appendix A.2.1.

LTL$_f$ is more expressive than Action-Set goals. Every Action-Set goal g_ϕ can be expressed as an LTL$_f$ goal. To do so, ϕ is transformed into disjunctive/conjunctive normal form and every A_i is replaced by a $\Diamond A_i$ and every $\neg A_i$ by a $\Box \neg A_i$.

In the following section, we evaluate the scalability of our algorithms with respect to the number of temporal goals and compare the LTL$_f$ and Action-Set compilation.

EXPERIMENTS SETUP & BENCHMARK

We implemented both, the LTL$_f$ and Action-Set compilation in the Fast Downward[9] translator [Helmert, 2009] after the grounding step. This process requires the grounded PDDL instance and a list of temporal soft goals as input, and outputs a modified task with single goal facts for each temporal soft goal. To generate the NFAs, we use a tool developed by Camacho [2017]. This tool provides a DFA that we pass to Spot (version 2.6.3) [Duret-Lutz et al., 2022] to transform the DFS into a complete NFA.

The temporal goals are specified in JSON format and are passed to our extension of Fast Downward as an additional input. For an example and the input definition, please refer to Appendix A.2.2.

Benchmark We use 7 domains as a benchmark[10]. Nomystery, Rovers R and TPP, as used by Nakhost et al. [2012] and, Blocksworld, all with two encoded resources for trucks/rovers/hands. Rovers T and Satellite are based on the IPC domains, which have been extended with encoded time windows for uploading data and capturing images. Lastly, Parent's Afternoon, a domain we developed for the user study (see Section 4.5). It models the afternoon activities of a family where a subset of the activities is constrained by time windows. All these domains have encoded constraints, i. e. resources or time, see Section 5.4 for a detailed discussion. Thus, we do not use a cost bound, but rather choose the constraints so tight, that not all soft goals are satisfiable.

Each problem has a fixed number of non-temporal goal facts (Blocksworld 8-13, Nomystery 5-8, Parent's Afternoon 4-8, Rovers R 5-7, Rovers T 4-8, Satellite 5-10, TPP 4-6). Additionally, we generate two sets: `onlyAS` and `all`, each with up to 15 temporal goal facts for every instance. The temporal goals are based on domain-dependent templates, which are listed in the Appendix in Table 16. `onlyAS` includes only temporal goals that can be

[9] `https://github.com/aibasel/downward`
[10] `https://doi.org/10.5281/zenodo.14988342`

encoded as Action-Set goals, while for `all` the selection is not restricted. The templates are instantiated by randomly selected objects. We then generate 15 benchmark instances with 1 to 15 temporal goals based on a fixed sequence. Only instances that contain at least one MUGS are included in the benchmark. All goals, temporal and non-temporal, are treated as soft goals. The scripts for generating the instances and the temporal goal in both encodings are publicly available[11].

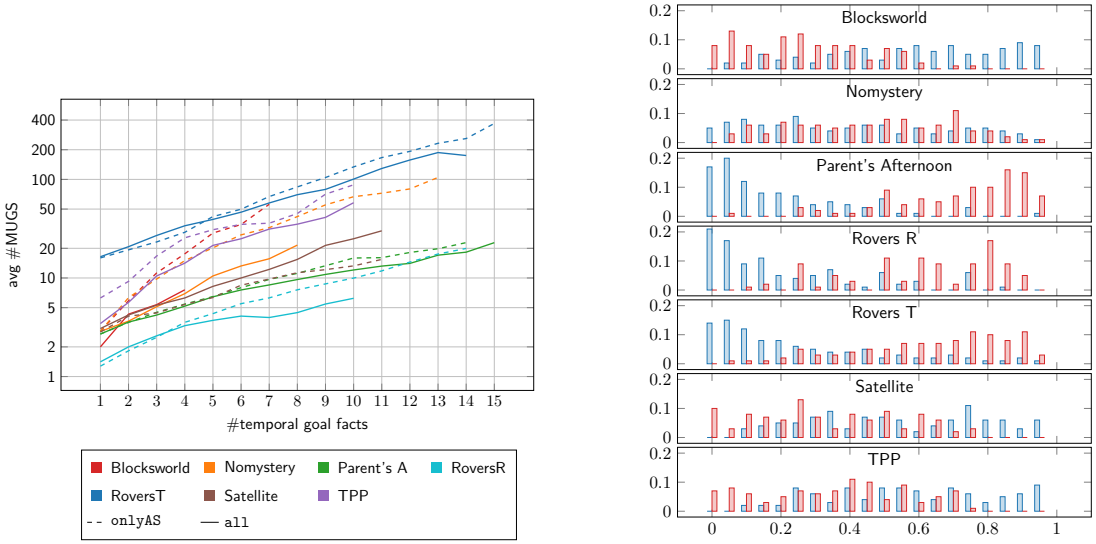


Figure 17: **Left**: Average number of MUGS over number of temporal goal facts per domain of instances solved by SysS. Only data for number of temporal goal facts with more than 9 data points are displayed. **Right**: Histogram over explored goal lattice fraction for SysS (blue) and SysW (red) in `onlyAS`.

The average number of MUGS over the number of temporal goals are depicted on the left in Figure 17. As expected, for all domains the number of MUGS increases with the number of temporal goals. However, the absolute numbers can vary significantly depending on the domain. The constraint levels of the domains, i. e. the position of the border between MUGS and MSGS in the goal lattice, affect the performance of algorithms, especially GLS. The fraction of the goal lattice explored by SysS and SysW is shown on the right in Figure 17. Nomystery has a more or less equal distribution. Blocksworld, Satellite, and TPP have more large MUGS, while Parent's Afternoon, Rovers R and Rovers T have smaller MUGS.

Algorithms In the following, we focus on the best-performing algorithms in the IPC domains. Thus, we use goal lattice search with the symbolic approach to test for solvability of each soft goal subset. Since it is not known in advance how constrained the instances are, we use both expansion orders systematic strengthening SysS and weakening SysW.

For GSBnB, h^{car} is not suitable for pruning because the way the individual abstractions are constructed, they favor good estimates for reachable states over proving unreachability. Since our benchmark has encoded resource and time constraints, rather than an external cost bound, only proof of unreachability, i. e. heuristic estimates of ∞ for individual goals,

[11] `https://doi.org/10.5281/zenodo.14989566`

will contribute to pruning. For h^{pot}, we consider a goal fact unreachable if its estimate is above 10^8. For more details about h^{pot} and dead-end detection we refer to [Seipp et al., 2015]. However, it turned out that h^{pot} is not suitable for instances with compiled temporal goals. For a majority of the tasks the resulting LP exceeded the memory limit. Thus, we only use $h^{\max}$ and the blind heuristic as a baseline.

OVERALL COVERAGE

First, we evaluate the overall coverage, the number of instances where the MUGS could be computed within the specified time and memory limits. The coverage results for `onlyAS` for both encodings and for `all` with the LTL_f encoding are given in Table 6 and Table 7 respectively.

	domain	GSBnB(blind)		GSBnB(h^{max})		SysS		SysW	
		AS	LTL_f	AS	LTL_f	AS	LTL_f	AS	LTL_f
onlyAS	Blocksworld(557)	**250**	**351**	247	308	173	168	169	165
	Nomystery(575)	125	103	206	165	**402**	**303**	295	245
	Parent's A(730)	395	419	531	470	**709**	**696**	546	539
	Rovers R (580)	211	289	247	298	**505**	**501**	296	295
	Rovers T (777)	531	746	377	422	**746**	**733**	475	475
	Satellite(596)	189	208	196	183	**332**	**324**	299	283
	TPP(340)	124	136	173	192	**240**	216	231	**220**
	sum (4155)	1825	2252	1977	2038	**3107**	**2941**	2311	2222

Table 6: Coverage results per domains of `onlyAS`, Action-Set (AS) vs. LTL_f encoding. The best performance within each algorithm is underlined, the best performance overall algorithms per encoding is shown in **boldface**.

	domain	GSBnB(blind)	GSBnB($h^{\max}$)	SysS	SysW
all	Blocksworld(545)	**322**	278	118	159
	Nomystery(558)	65	120	**240**	228
	Parent's A(847)	383	498	**701**	509
	Rovers R (561)	260	272	**354**	294
	Rovers T (797)	**765**	445	679	461
	Satellite(620)	258	248	**343**	299
	TPP(394)	202	253	**260**	254
	sum (4322)	2255	2114	**2695**	2204

Table 7: Coverage results per domains of `all` encoded as LTL_f goal. The best performance overall algorithms is shown in **boldface**.

Overall, the best approach is SysS, followed by SysW. Comparing the coverage between Action-Set and LTL_f encoding within each algorithm shows, that only blind state space search prefers the LTL_f encoding, with $h^{\max}$ it is domain dependent and goal lattice search (GLS) performs better with Action-Set encoding. With the preferred encoding SysS performs best

in 6 domains and GSBNB$(blind)$ in 1 for `onlyAS`. For `all` SysS solves most instances in 4, GSBNB$(blind)$ in 2 domains. Comparing GSBNB$(blind)$ and GSBNB$(h^{\max})$ shows that in 4 domains, Nomystery, Parent's Afternoon, Rovers R and TPP, using pruning based on $h^{\max}$ can improve the performance significantly. SysS performs significantly better than SysW in Parent's Afternoon, and the Rovers domains due to the smaller fraction of the goal lattice that needs to be explored (see right Figure 17).

SCALING OVER NUMBER OF SOFT-GOALS

Next, we evaluate how many temporal goals are feasible. The coverage over the number of temporal goals for `onlyAS` and `all` are depicted in Figure 18 and Figure 19 respectively.

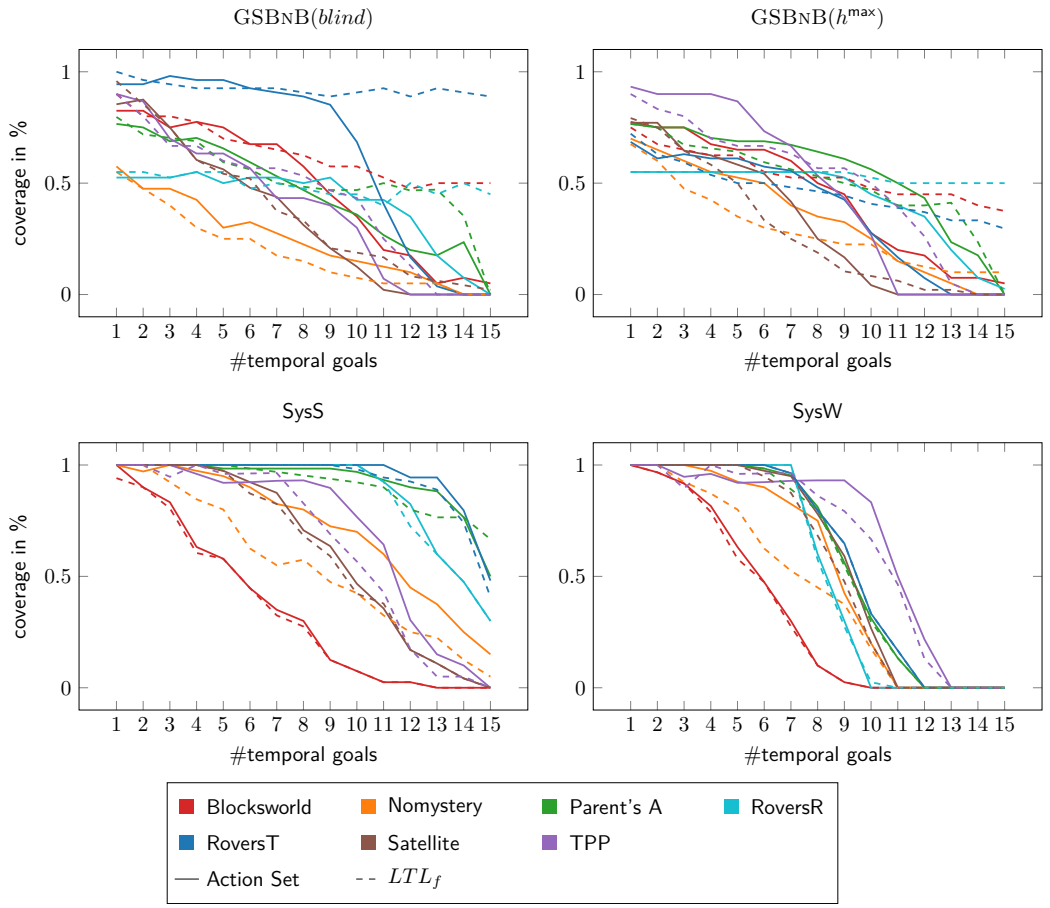


Figure 18: Relative Coverage over the number of temporal goals for `onlyAS`, Action-Set encoding dashed, LTL_f encoding solid lines.

For GSBNB$(blind)$ and GSBNB$(h^{\max})$, the decrease in solved instances is similar across all domains, except for Rovers T. For most domains, there is no significant difference between Action-Set and LTL_f encoding for up to about $8-10$ temporal goals. For more temporal goals, the LTL_f encoding has a clear advantage. For SysW, the coverage for most domains with increasing numbers of goals only decreases slightly or not at all, until a sudden drop. For SysS, this behavior is delayed until more soft goals are handled. For GLS there is no advantage for the LTL_f encoding for larger number of temporal soft goals.

Overall, the most consistent advantage of the Action-Set encoding is in Nomystery.

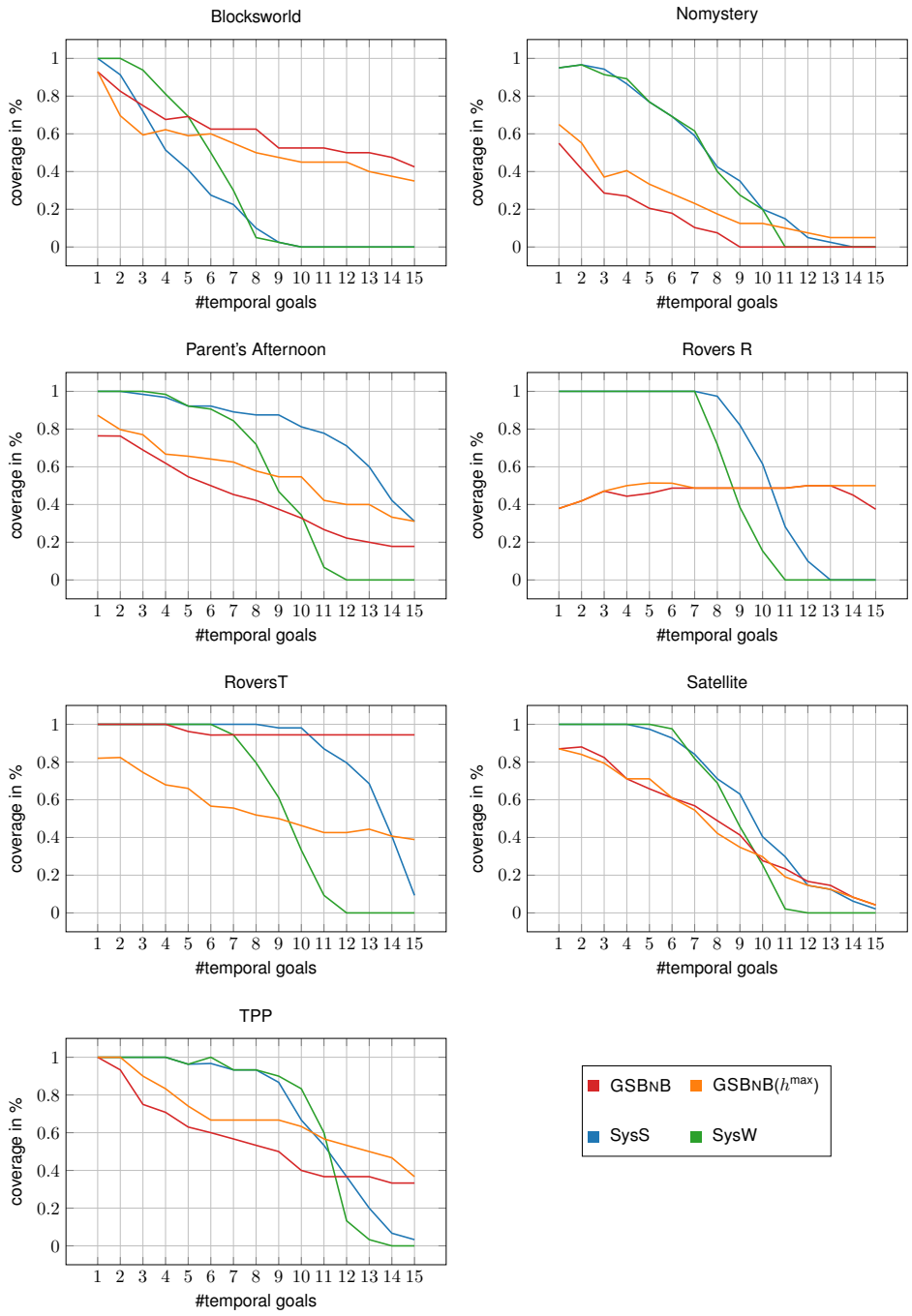


Figure 19: Relative Coverage over the number of temporal goals for `all`.

Figure 19 depicts the coverage comparison per domain on the `all` benchmark set. GLS outperforms GSBNBconsistently for smaller number of soft goals. In Blocksworld GSBNB performs better with 7 temporal goals, in Rovers R with 11 and in TPP and Rovers T with 12. However, in Nomystery, Parent's Afternoon and Satellite GSBNB does not surpass GLS.

PERFORMANCE ANALYSIS

To analyze the reason for performance differences depending on the encoding, we compare the number of expansions and the search time for both encodings in Figure 20 for the branch-and-bound approaches and in Figure 21 for GLS.

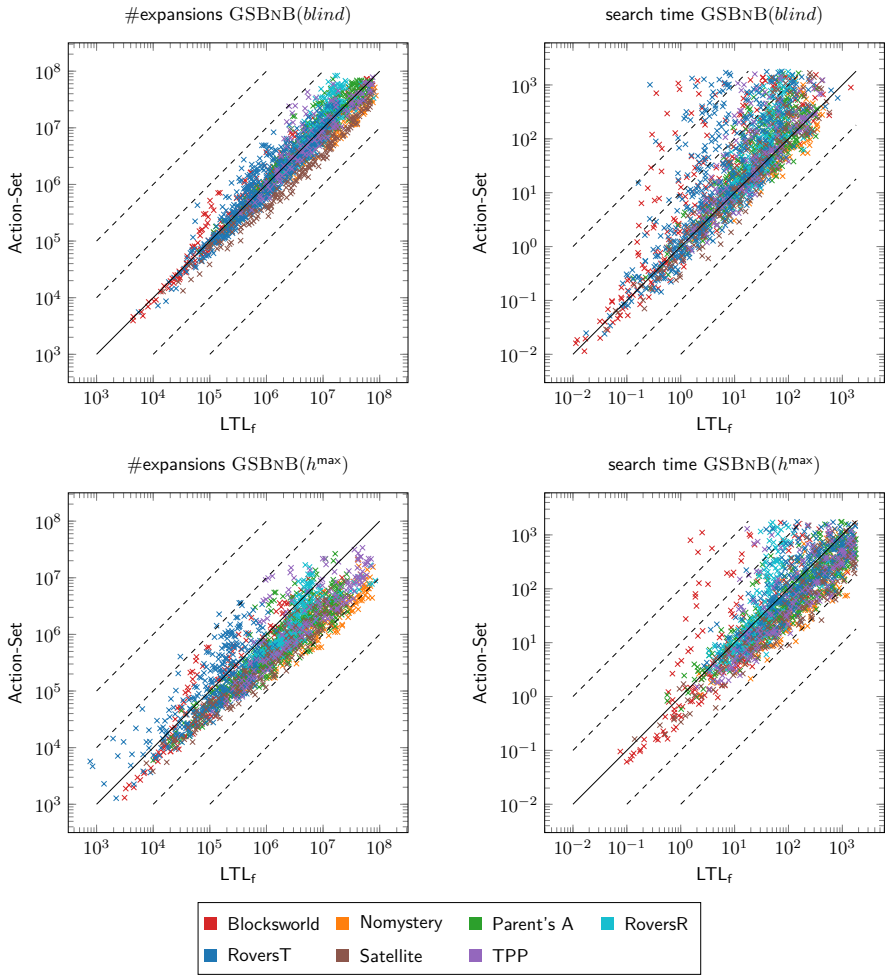


Figure 20: Instance wise comparison of number of expansions and search time for `onlyAS`, between Action-Set encoding (y-axis) vs. LTL_f encoding (x-axis)

It is important to note that the state space and thus search space sizes of the two compilation approaches differ. For the LTL_f encoding, there is one additional expansion for each temporal goal due to the automaton synchronization. However, the order in which the automata are evaluated is fixed. For the Action-Set encoding, there are additional states during the planning phase, depending on the use of an action set, but no additional actions. The new actions are limited to the evaluation phase. Since the evaluation order of the temporal goals is not fixed, a larger number of them leads to a larger branching factor.

For GSBnB$(blind)$, the number of expansions is comparable for both encodings. The larger branching factor for the Action-Set encoding leads to a higher search time, particularly in domains with many MUGS, Rovers T and Blocksworld. For GSBnB(h^{max}), except for some instances mostly Rovers T and Blocksworld, the number of expansions is up to one

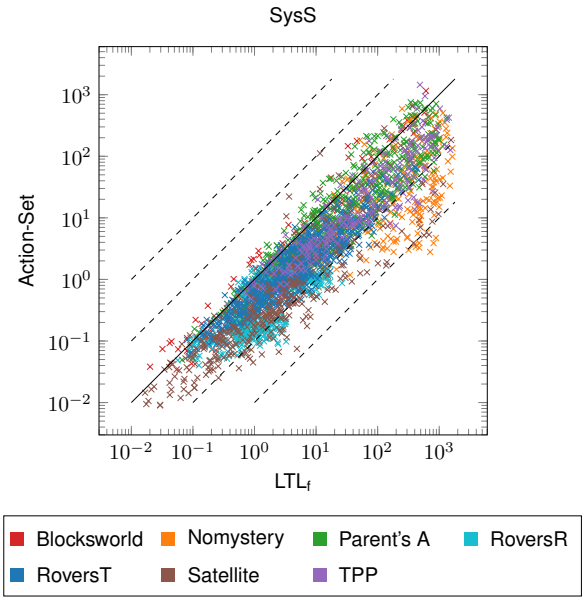


Figure 21: Instance wise comparison of time to constructed BDD for reachable search space for `onlyAS`, between Action-Set encoding (y-axis) vs. LTL$_f$ encoding (x-axis)

order of magnitude larger for the LTL$_f$ encoding, indicating, that $h^{\max}$ performs better with the Action-Set encoding. The smaller search space is also reflected in a shorter search time, exceptions are some instances from Blocksworld and Rovers R. This shows that the Action-Set encoding is overall more beneficial for $h^{\max}$.

For GLS we compare the time it takes to generate the BDD representing the reachable state space. The LTL$_f$ encoding takes overall up to 1 order of magnitude and for Nomystery specifically up to 2 orders of magnitude longer. This clearly shows why Action-Set encoding is more suitable for GLS.

4.4 Iterative Planning Platform

We implemented the web-based platform IPEXCO for Iterative Planning with Explanations of COnflicts. It enables users to perform iterative planning with temporal soft goals and provides explanations based on soft goal conflicts. The platform also supports unsupervised online user studies by accommodating lay person, in addition to providing tools for defining test instances and evaluating the performance of test persons in a controlled environment.

In the following, we first describe the workflow supported by the tool, and then the extensions made to conduct user studies.

4.4.1 Iterative Planning Workflow

First we describe the full workflow supported by the tool. The target group of users is people familiar with planning. The adaptations for a layperson taking part in a user study are described in the next section.

Initially, the user provides a planning task. The domain and initial state are provided

as PDDL [Fox and Long, 2003] domain and problem file. Both are fixed and cannot be changed during the iterative process. Hard and soft goals are defined separately and can be changed for each iteration step, reflecting the evolution of user preferences. A sample collection of goals is depicted in Figure 22.

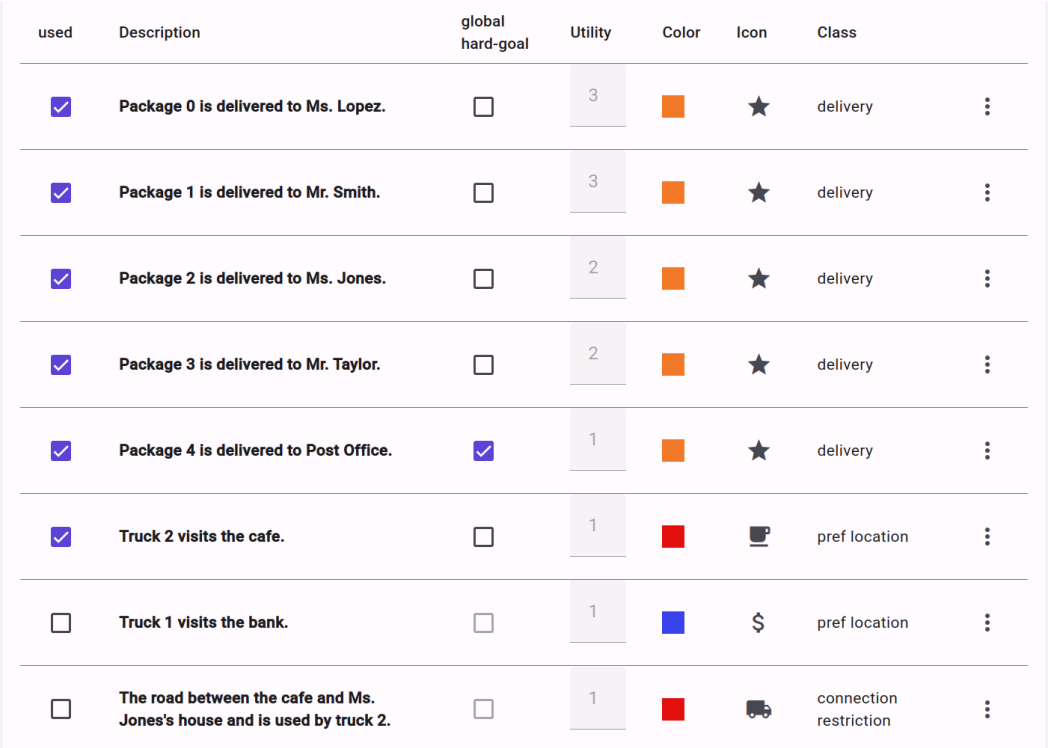


Figure 22: A sample collection of goals as displayed by the platform. Each goal can be associated with an integer utility and a color, icon and a class which can be used for a visual grouping and reference. Global hard goals must be satisfied in each iteration step.

Goals are defined by explicitly stating the corresponding goal fact, Action-Set, or LTL_f formula. In addition, there exists the possibility to use domain-dependent templates. These map a predefined natural language representation to the corresponding formula. The user then only needs to select objects, pre-selected by type and further constraints, for each parameter. An example of using such a template to create a goal for road usage is shown in Figure 23. In this example, the location selection is restricted by their type and whether they are connected. In the appendix, we provide sample templates in Table 16, and describe the definition of such a template for IPEXCO in Section A.3.

Figure 23: Interface for goal definition based on natural language definition.

The iterative planning process is divided into individual steps. In each step, if possible, a sample plan is provided. Based on this sample plan, the user decides which goals to enforce in the next iteration step. While doing so, the user has access to the explanations

based on the current sample plan. Figure 24 depicts a diagram visualizing the interaction between the user and the platform within one iteration step. Figure 25 shows the main interface for iterative planning implementing these interactions.

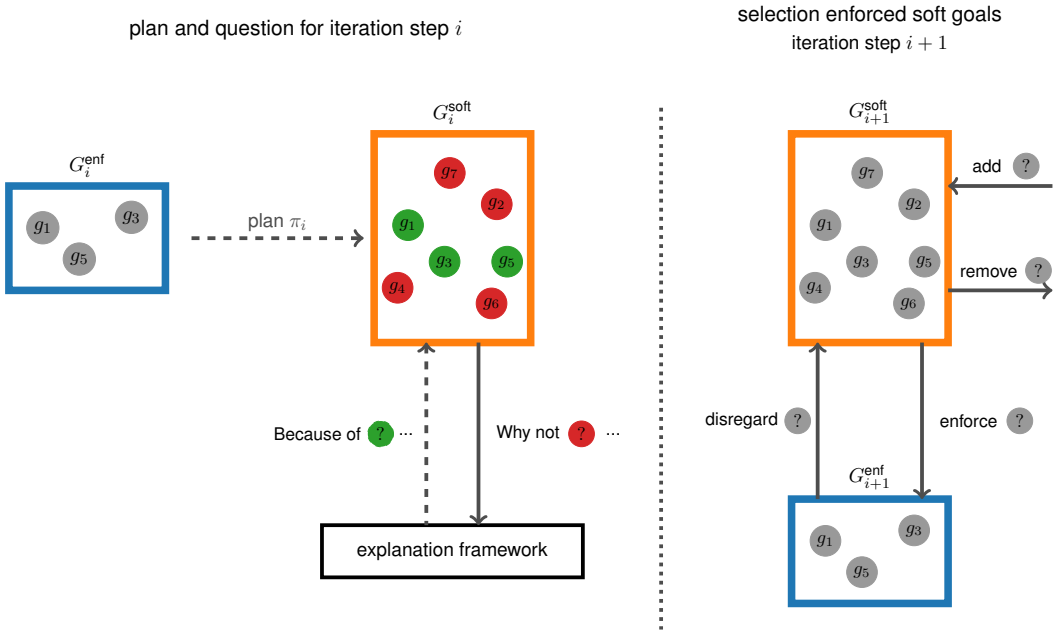


Figure 24: Diagram of supported iterative planning workflow with explanations for iteration step δ_i in case a plan exists. $G^{\text{true}}(\pi_i)$ are represented by green circles and $G^{\text{false}}(\pi_i)$ by red ones. Dashed arrows represent actions performed by the platform; solid arrows reflect the interaction possibilities of the user.

At the beginning of each iteration step δ_i a plan π_i, if possible, is computed for task $\tau_i = (V, A, c, I, G^{\text{hard}} \cup G_i^{\text{enf}}, G_i^{\text{soft}} \setminus G_i^{\text{enf}}, b)$. Depending on whether there exists a plan, the user is either provided with the soft goals satisfied $G^{\text{true}}(\pi_i)$ and not satisfied $G^{\text{false}}(\pi_i)$ by plan π_i (see column 2 of Figure 25) or with the fact, that there is no plan. In either case, the user must decide which soft goals G_{i+1}^{enf} should be enforced in the next iteration (see column 4 of Figure 25). They are not restricted to the existing soft goals G_i^{soft}. They can add new soft goals that they did not think of initially or that they did not consider relevant, and they can remove soft goals that are no longer needed. This is done through a separate interface shown in Figure 22.

While deciding which soft goals to add and enforce, the user can ask questions (see column 3 of Figure 25). Note, that we decided to only consider the soft goals G_i^{soft} in the questions and answer in step δ_i. If there is no plan, the only available question is "Why is τ_i unsolvable?". The answer according to Definition 16 lists the MUGS that are part of G_i^{enf}. If there is a plan, then the user can ask "Why is Q not satisfied?", where $Q \subseteq G^{\text{false}}(\pi_i)$. Then based on Definition 15, an answer lists all subsets of $G^{\text{true}}(\pi_i)$ that can no longer be satisfied if Q would be enforced. Since the soft goals can change between iteration steps, we use the task transformation (Theorem 2) to provide the individual answers.

The explanation interface itself works as follows. To ask a question, the user selects a soft goal g that is not satisfied by the current plan, as shown in Figure 26. The selection

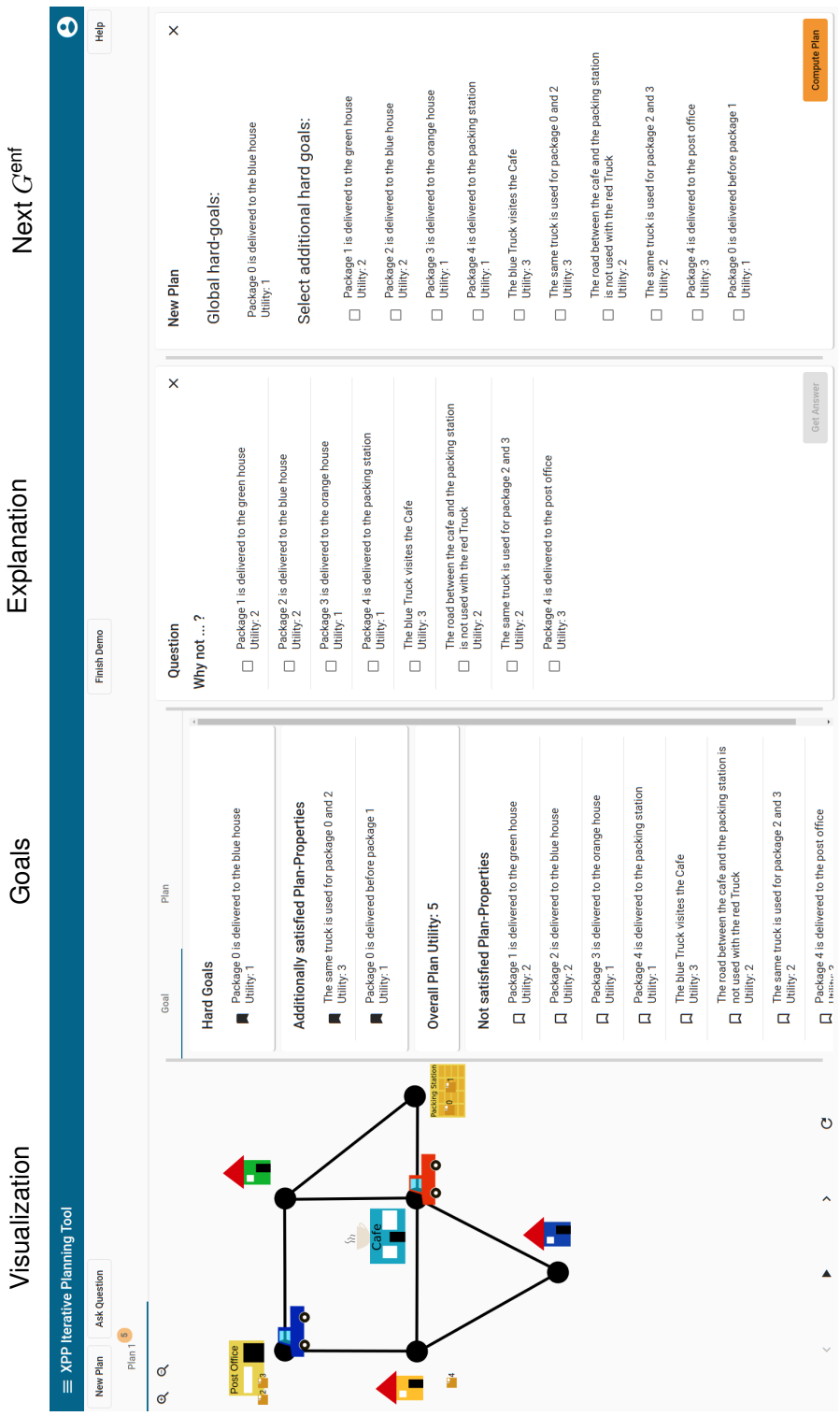


Figure 25: Main interface for iterative planning with explanations; first column: graphical representation of the planning task; second column: goals satisfied and not satisfied by the current plan; third column: explanation interface; fourth column: selection of enforced goals for the next iteration.

is interpreted as the question: “Why is g not satisfied by the current sample plan?”. The answer is provided as a list of the soft goal subsets as depicted in Figure 27 on the left. In the depicted shown, the answer states: “If you deliver package 4 to the post office, then you cannot deliver package 1, and you can either not deliver package 2 or you cannot use the same truck for packages 2 and 3”. If the selection of enforced goals is unsolvable, the user can ask *why* it is unsolvable. In the example explanation, shown in Figure 27 on the right, the enforced goals are unsolvable because: “You cannot deliver package 2 if you have to use the same truck for packages 0 and 2”.

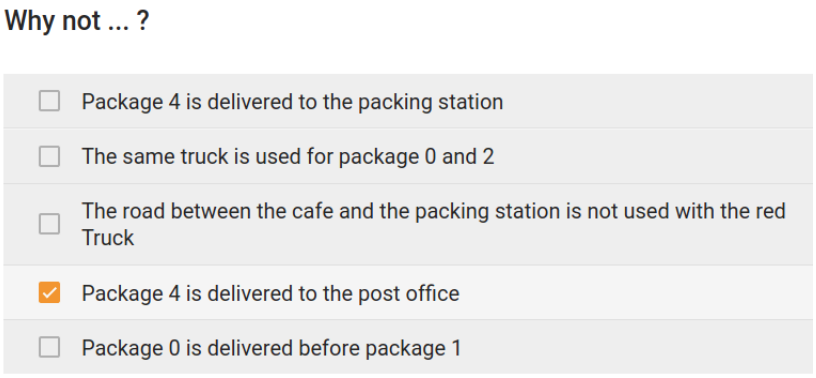


Figure 26: Interface to ask questions. The user selects on goal g from the list of unsatisfied goals. This selection is interpreted as the question: “Why is g not satisfied by the current sample plan?”

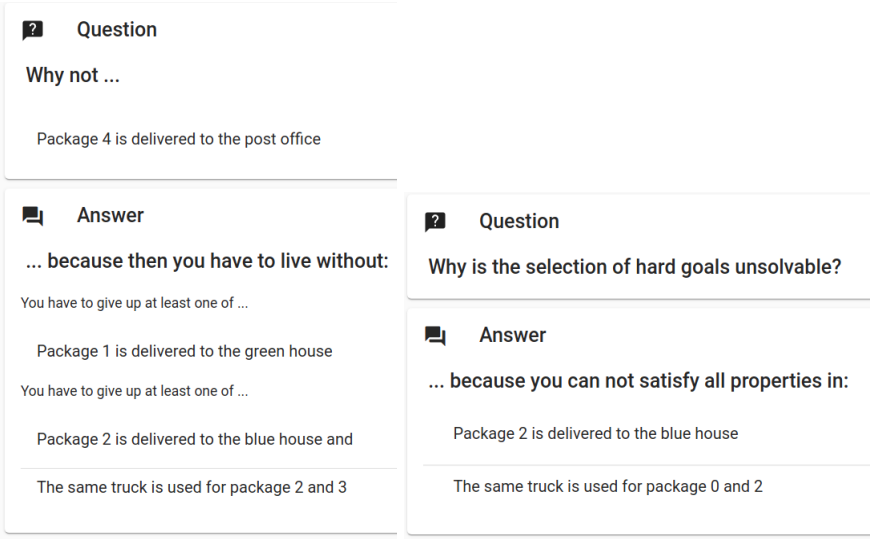


Figure 27: Left: Answer to question in Figure 26. The answer is given as a list of goal subsets. From each subset the user must forego one goal to allow the goal in the question to be satisfied. Right: Example answer to an unsolvable set of enforced goals. The answers list all MUGS that are a subset of the enforced goals.

The user can ask as many questions as necessary. Once they commit to a new set of soft goals G^{soft}_{i+1} and a new set of enforced goals G^{enf}_{i+1}, the next iteration step starts.

This outlines the whole supported iterative planning process. The adaptations for layperson and restrictions for user studies is described next.

4.4.2 Adaptations and Extensions for User Studies

The platform includes special adaptations for laypersons and supports unsupervised online user studies. First, we focus on the restrictions of the iterative planning workflow and accommodations for laypersons.

In order for a task to be utilized in a user study, the study creator must define a fixed set of soft goals G^{soft}_{fix}. This allows for the pre-computation of the MUGS, enabling instantaneous responses and ensuring a responsive experience for test persons. Additionally, this ensures that all test person are evaluated on the same tasks with the same set of soft goals.

Especially in user studies with laypersons, it is essential that the task and goal description is as accommodating as possible. Each task can be accompanied by a visualization of for example the map structure. Additionally, goals are not depicted as LTL_f formulas but rather in natural language, specific to the individual instance. Some examples can be found in Figure 25. The user study creator is responsible for providing all task-specific visualizations and natural language descriptions.

In many domains only domain experts have intrinsic preferences for different soft goals. For non-expert test persons, it is necessary to provide artificial preferences. To this end, a utility, defined as a fixed integer value, was added to each goal. The utility is assigned in the list of soft goals in Figure 22 in column four and is visible below each goal as illustrated in Figure 25.

Online User Study Support To support unsupervised online user studies, additional extensions are necessary. To recruit and pay test persons, the platform connects to the online recruitment service Prolific[12].

To ensure a comprehensive evaluation, it is essential to closely monitor the test participants' actions, recording each action with a timestamp. This includes all asked questions, and enforced soft goals, enabling the reconstruction of the timeline of the iterative planning process.

A user study is not limited to a single iterative planning process for a particular task, it can consist of multiple tasks from different domains. It is possible to designate certain tasks as *training tasks*, which provide instructions to facilitate user familiarization with the tool. In addition, user studies can be enriched with information for test persons, such as the course of the user study or descriptions of the tool and domains. The incorporation of external questionnaires is also a possibility.

4.5 User Study Evaluation

We have demonstrated, that allMUGS can be solved reasonably efficiently. However, we have yet to investigate whether the resulting explanations are useful for users. In the

[12] www.prolific.co

following section, we present a user study that evaluates this question in terms of test person performance in different case studies on iterative planning.

The long-term target audience for our explanations is domain experts who have intrinsic preferences and have experience with the task at hand. The aim of the iterative planning process with explanations is to support users in refining their preferences and understanding the dependencies between them, enabling them to shape the plan accordingly and identify a compromise. Ideally, the test persons would be experts in the individual case study domains. Unfortunately, for a number of reasons including the abstract and non-applied nature of the technology being evaluated, it is difficult to recruit these experts for XAIP baseline research. However, when evaluating the usefulness of conflict explanations for iterative planning in general, there is no need for expert users. We therefore decided to conduct an online user study with crowdworkers recruited via Prolific [Palan and Schitter, 2018]. This allowed a sufficiently large number N of test persons to obtain statistically significant results.

We conducted a study on three different planning domains, including a new domain "Parent's Afternoon", which encodes the common challenge of family logistics, familiar to layperson users. For each domain, we developed a use case with conflicting soft goals that are complex enough to be non-trivial, yet simple enough to be solved within the limited amount of time that crowd workers are willing to invest. We used $N = 100$ test persons for each domain. To assess the impact of the explanations on the test person performance they were divided into two groups of equal size: one with access to explanations and one without.

4.5.1 Case Study Design – Planning Domains and OSP Tasks

To cover different sources of conflicts between soft goals, we used three different domains. Other XAIP user studies used one domain [Chakraborti et al., 2019b, Chakraborti and Kambhampati, 2019, Sreedharan et al., 2019c, 2020c, Lindsay et al., 2020, Das et al., 2021, Kumar et al., 2022, Das et al., 2023, Shvo et al., 2022], two domains [Sreedharan et al., 2019b, 2020d, Brandao et al., 2021b], three domains [Brandao et al., 2022a] or four domains [Krarup et al., 2021].

We introduce a new domain, called *Parents Afternoon*, that encodes family logistics familiar to lay users for example driving children to sports events and grocery shopping. The source of conflicts are pick-up/drop-off/opening times and the limited load capacity of the car. Our two other domains, Transport and Rovers, are variations of the IPC domains Nomystery and Rovers. Transport encodes the transportation of packages via multiple trucks on a road map with fuel consumption as a constraint. Thus, soft goals compete for the same consumed resource, namely the limited fuel of the trucks. Competition for resources is a ubiquitous source of conflict, consider for example money. This structure is also natural, and should be familiar to lay users to some extent. Rovers encodes data collection and transmission on Mars, which is constrained by both resource consumption and time constraints for uploading data. This combines the two conflict sources mentioned above. This specific problem is unknown to the layperson, but the underlying conflict-inducing structure is natural and easy to understand.

The complexity of the domain instances, i. e., the OSP tasks, requires careful attention.

The task and soft goal conflicts must be sufficiently complex to be interesting, yet practicable for users in crowd-sourcing. If the task is too complex or too long, test persons tend to abandon it quickly, resulting in data that is not representative of real interaction with the system. Given the exponential nature of the underlying structures (the size of the state space and the number of MUGS), the transition from too easy to too hard instances is fast, and the ideal task complexity is on a narrow edge. Our OSP task design balances the difficulty level as follows.

We keep the soft goals simple so that they are easy for laypersons to understand, and it is easy to see whether a particular soft goal is satisfied by the current plan. On the other hand, MUGS of size 2 tend to be easier to identify and to remember, so we designed our tasks to mostly feature larger MUGS incorporating more complex conflicts. Similarly, we have tried to avoid soft goals, which included a large part of MUGS. Eliminating one of these goals would immediately resolve most conflicts and thus make the task too easy.

We fine-tuned the task size and MUGS complexity based on small test studies, and we settled on the instances[13] depicted in Figure 28. More details are given in the Appendix in Section A.4.

Parents Afternoon The Parent's Afternoon instance has 6 locations and 4 persons, items and activities. We defined 13 goals, reflecting achieved activities and ordering relations between those.

1. Shopping is done.
2. Grandma's shopping is done.
3. Parent's sports is done.
4. Soccer training is done.
5. Music Lesson is done.
6. Bring friend to sport center.
7. Kid1 is back home.
8. Kid2 is back home.
9. Groceries are at home.
10. Grandma's groceries are at grandma's house.
11. Grandma is back home.
12. Shopping is done before sports.
13. Grandma and friend are not together in the car.
14. Shopping is done before sports.

The instance has 25 MUGS (size 2: 3, size 3: 13, size 4: 7, size 5: 11, size 6: 1). One example MUGS is $\{$"Bring friend to sport center", "Grandma's shopping is done", "Shopping is done before sport"$\}$.

Transport The Transport instance has 9 locations, 2 trucks and 5 packages. There are 13 goals reflecting the delivery of packages, use or non-use of road connections, location visits, and ordering relations between packages. There are 37 MUGS (size 3: 14, size 4: 7, size 5:

[13] https://doi.org/10.5281/zenodo.14988342

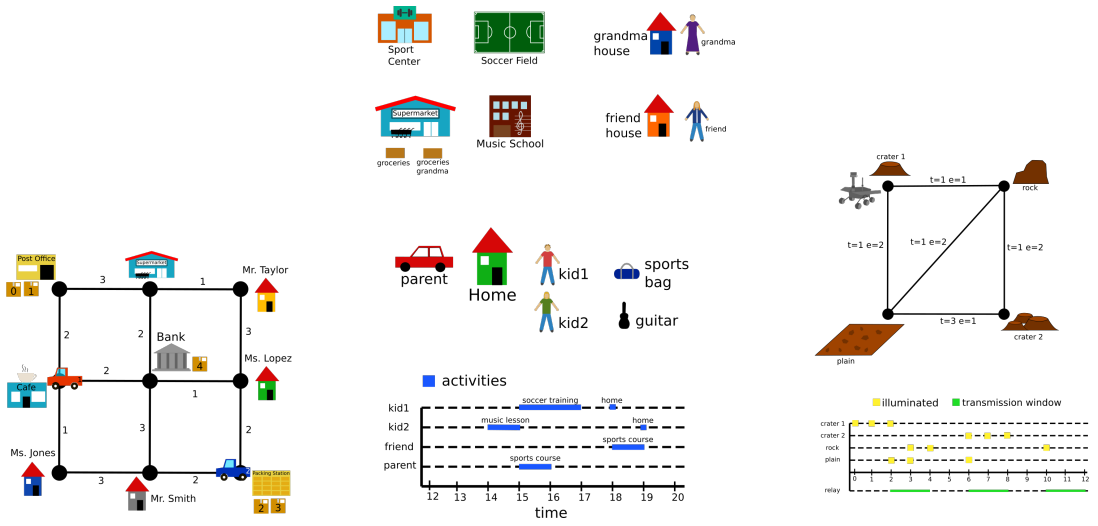


Figure 28: Visualization of the three instances used in the user study, from left to right: Transport, Parent's Afternoon and Rovers.

2, size 6: 2). One example MUGS is {"P_0 is delivered", "P_1 is delivered", "Supermarket is not visited with truck 2"}.

Rovers The Rovers instance has 1 rover, 4 locations and 10 tasks. We designed 14 goals pertaining to task achievement and the order of data uploads. There are 102 MUGS (size 2: 22, size 3: 56, size 4: 4). One example MUGS is {"Uploaded rock image", "Uploaded crater 1 X-ray image", "Crater 2 image is uploaded before rock sample"}.

4.5.2 User Study Design

After clarifying the test instances, we now turn to the user study itself. We begin by discussing user motivation and then cover the general setup including test-person recruitment, payment, and the experiment workflow.

User Objective

In application scenarios of iterative planning, users aim to understand conflicts between soft goals and to converge to an acceptable compromise. However, in a user study, test persons lack intrinsic motivation to do so. Therefore, we provide them with an objective to pursue, namely additive utility maximization. This objective, canonical as it is, is easy to understand for layperson users. We assign a fixed utility to each soft goal, and the test persons are tasked with finding a solvable selection of soft goals, that maximizes the summed utility. Fixed utility is a standard form of oversubscription planning, which could be solved optimally using known algorithms (e. g. [Smith, 2004, Domshlak and Mirkis, 2015, Katz et al., 2019]). Nevertheless, this setup is meaningful for evaluating our explanation approach, as test persons in our study need to understand the conflicts to perform well.

In what follows, keep in mind that this objective is *only in the heads of the test persons*. Neither the planner for the sample plans nor the explanation method take it into account.

This reflects the targeted application scenarios, where no such fixed (and simple) objective exists.

The objective is linked to payment via a bonus that increases with the utility achieved, providing a strong incentive to find good plans. Some prior work, e. g. [Chakraborti et al., 2019b], has followed similar schemes. The basic compensation for participating in the user study is $5£$, the maximum reachable bonus payment is $2.50£$. As shown in Figure 29 we include a progress bar to convey to users the maximal possible utility, the utility that has been reached, and the bonus payment levels. In preliminary test runs, we found that such explicit information helped to motivate test persons.

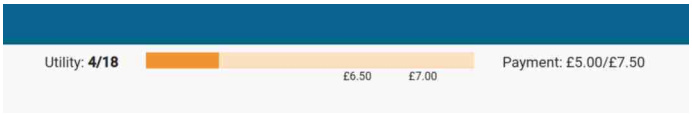


Figure 29: Progress bar indicating the maximal possible utility, the utility that has been reached and the levels to reach for bonus payment.

User Study Setup

To evaluate the impact of the explanations, we divided test persons randomly into groups with vs. without the option to ask questions. We refer to these two groups as `Q+` and `Q-` respectively.

We fixed the ordering of domains to Transport, Parents Afternoon and then Rovers. To maximize familiarity of test persons with the IPEXCO tool and iterative planning, we re-invited test persons to also address the remaining domains. We waited with domain i until the user study on domain $i-1$ was completed, to maximize the number of re-invited test persons. Finally, we fixed each person's assignment to the `Q-`/`Q+` group across all domains. This serves to obtain consistent streams of test persons who are increasingly familiar with one of the two tool variants. Distributing re-invited users across both variants would have resulted in too many different subgroups for a meaningful analysis.

In addition to user performance measurements, we included a questionnaire for subjective measures. We used a Likert scale from 1 to 7 to measure the test person's opinions. The questions are listed in Table 9, which is included in our result's analysis (Section 4.5.3) for ease of reading.

Beyond these fixed-answer questions, we also included free-text questions, targeted at qualitatively assessing the presentation and usefulness of explanations in the proposed setting. The `Q+` test persons were asked for comments about the structure and presentation of the questions and explanations. The `Q-` test persons were asked to list questions that would have helped them to completing the task.

Each experiment, i. e. each test-person run addressing the OSP task from one of our domains, proceeded according to the following workflow:

1. **Textual domain description**: A general description of the domain is provided, including an explanation of the possible soft goals and their varying levels of utility. The

constraints present in the domain are highlighted, and their impact on the satisfiability of the soft goals is addressed.

2. **Textual tool description**: The test persons are introduced to the iterative approach of the tool. Their objective of finding a plan with the maximum utility is emphasized. An instruction manual for the tool, accessible at all times, is provided. For Q+, the functionality of the questions is explained.

3. **Familiarization with tool through introductory instance**: Given the complexity of the task and tool, the test person is familiarized with the domain and tool through a small introductory instance. Test persons must compute at least one plan, and in Q+, they must ask at least one question before advancing to the next step.

4. **Planning for evaluation instance**: The test person processes the evaluation instance (as described in Section 4.5.1). The test person can exit the task at any time, particularly without achieving maximal utility.

5. **Questionnaire**: Finally the test person has to answer the questionnaire.

We used the test person recruitment facilities of Prolific [Palan and Schitter, 2018]. We applied two filters on test persons to obtain meaningful results. We required fluency in English and that at least 50% of each test person's previous submissions in Prolific must have been accepted by the respective study organizers.

4.5.3 User Study Results

In the following, we will evaluate the results of our user study. First, we address the test person statistics. Second, we present our main results, regarding the impact of the explanation facility on performance, in terms of utility achieved over time. Third, we analyze the impact of tool and task experience, comparing new vs. re-invited test persons. Finally, we evaluate the questionnaire results, in terms of a statistical analysis of the Likert scale answers, and in terms of a thematic analysis of free-text answers.

Test Person Statistics

	Transport		Parent's A.		Rovers	
	Q−	Q+	Q−	Q+	Q−	Q+
new	52	66	14	42	26	31
filtered out	-2	-16	-2	-11	-1	-4
re-invited	–	–	38	39	25	25
filtered out	–	–	0	0	0	-2
$\sum$	50	50	50	70	50	50

Table 8: Distribution of new, re-invited, and filtered-out test persons per domain and group.

As listed in Table 8, a total of 92 (139) individuals participated as Q− (Q+) test persons. 5 (31) of these test persons were filtered out because they did not complete the user study in

a meaningful way. Within Q+, we also filtered out those test persons who did not use the explanation facility, that is, who did not ask any questions.

For each domain, we continued running the study until we had 50 test-person runs for each of Q+ and Q−, resulting in a total of $N = 100$. For Parents Afternoon Q+, we increased the number of test persons to 70 in ensure a similar number of test persons and facilitate a meaningful comparison between Q+ new and Q+ re-invited. About 75% of the test persons participated in two domains, while 50% of the test persons participated in all three domains.

The time invested by the test persons in processing the evaluation instance is depicted on the left in Figure 30. With the exception of Transport, where more test persons from group Q+ stayed up to 15 minutes, Q− and Q+ exhibited similar behavior. Overall, half of the test persons spent up to 12min.

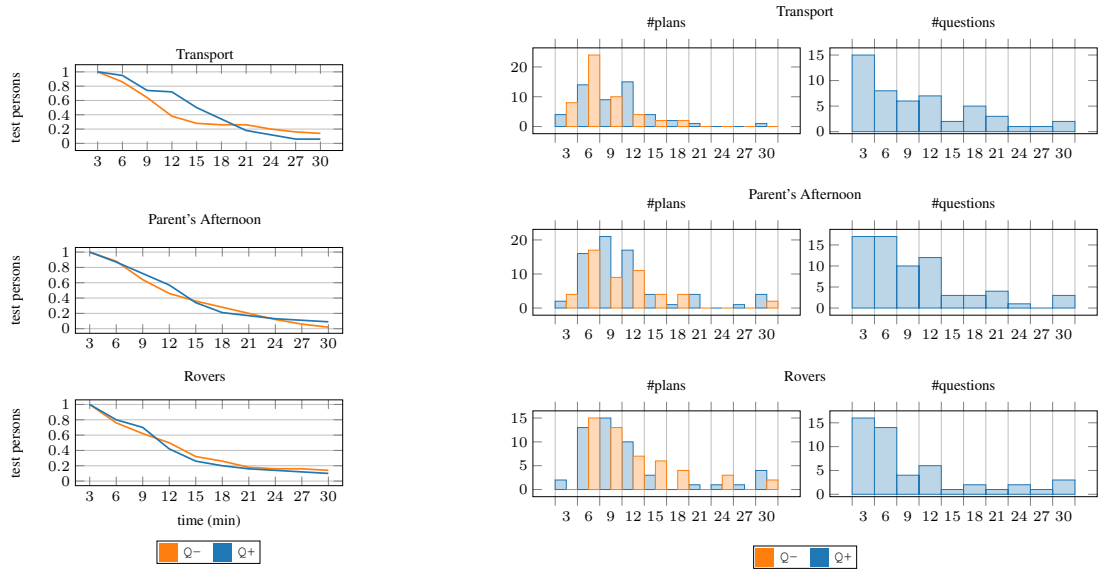


Figure 30: General Statistics: (left) Test persons over processing time: x-axis time in min; y-axis relative number of test persons that stayed up to that time. (right) Histogram of number of plans and questions: buckets of size 3, the label refers to the upper bucket border $[x, x + 3)$; y-axis number of plans/questions.

The histograms on the right in Figure 30 present the number of sample plans requested and for Q+ the number of questions asked. Overall, Q− and Q+ generate a similar number of plans, usually between 6 and 12 per task. The number of questions is relatively small, typically less than 12 which is approximately one question per plan.

USER PERFORMANCE FOR Q− VS. Q+

The main focus of our evaluation is to access the impact of explanations on performance, measured in terms of utility achieved over time. In what follows, we use the *Student's t-test* to determine the statistical significance of the difference between means, and the *Wilcoxon rank-sum test* for the difference between medians.

The iterative planning process on the evaluation instance required up to 30 minutes (with the remainder of the time allocated to the other components of the test-person workflow).

Figure 31 shows the utility as a function of this processing time.

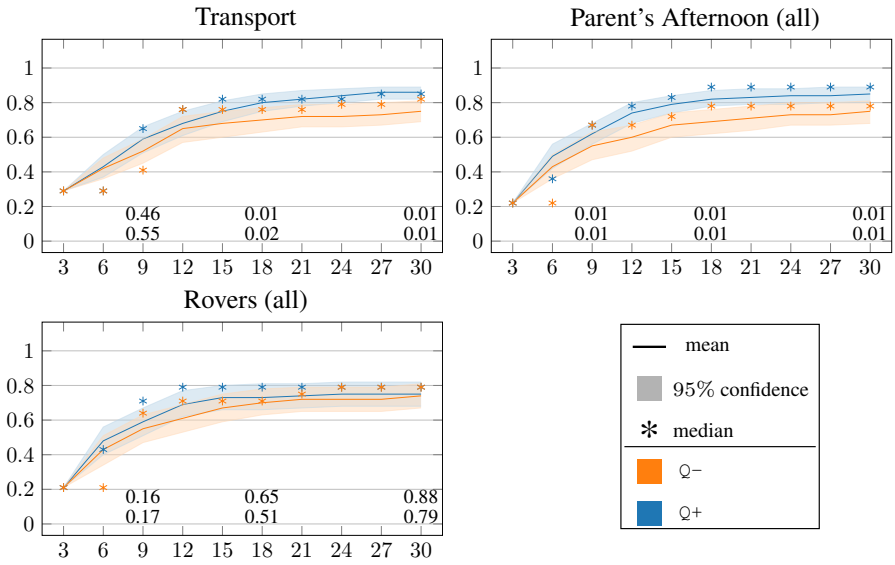


Figure 31: Performance over processing time: x-axis time in min; y-axis maximal achieved plan utility until that time. Numbers shown below the curve are p-values, i. e., the likelihood of the null hypothesis, for mean (top) and median (bottom) after 9/18/30 min processing time.

In all three domains, the mean utility for Q+ is higher than that for Q− across the entire timeline, indicating that Q+ indeed yields a performance advantage over Q−. In Parent's Afternoon, this advantage is statistically significant along the entire timeline of the experiment. In Transport, both groups initially make similar progress, but the Q− utility growth slows down earlier on while Q+ users are still gaining deeper insights and hence better trade-offs. Consequently, the advantage of Q+ over Q− is statistically significant for $t \geq 15$min. In Rovers, the timeline effect is inverse, with Q+ users initially making quicker progress but Q− users catching up eventually. For the majority of the timeline, the two curves are closer to each other than in the other two domains, and the advantage of Q+ over Q− is not statistically significant.

A deeper investigation of Rovers revealed that tool experience is a major factor here, more than in the other domains. This is presumably due to the more complex structure (resource consumption *and* time windows) and the larger number of MUGS. Figure 32 evaluates the impact by differentiating between new and re-invited users. In Parent's Afternoon re-invited test persons perform initially ($t < 15$min) better. This suggests that tool expertise can help achieve better utility faster. Given the smaller sets of test persons in both evaluations in Figure 32 and the substantial differences between individual crowd workers, the variance is quite high (compare to Figure 31), so statistical significance is found rarely. Nevertheless, in both evaluations, the differences between means and medians are consistent across the entire timeline.

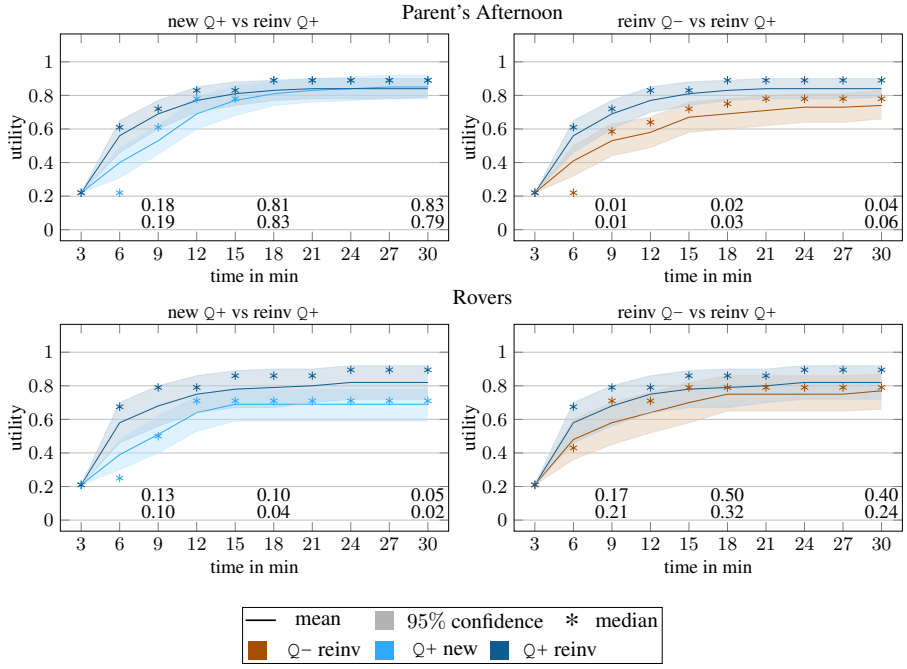


Figure 32: Performance over processing time in Parent's afternoon (top) and Rovers (bottom): (left) Q+ new vs. re-invited; (right) re-invited Q+ vs. Q−. p-values for mean (top) and median (bottom) after 9/18/30 min processing time.

WORK PROCESS Q+

Let us briefly analyze the work process of Q+ test persons. Figure 33 shows data for the number of questions asked, and plans generated, as a function of processing time.

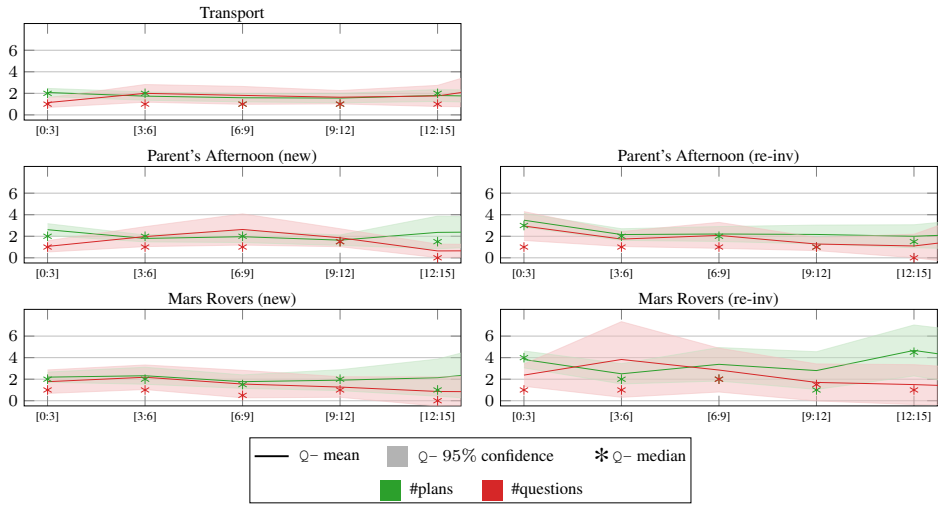


Figure 33: #plans and #questions over processing time: x-axis time in min (time buckets of 3 min); y-axis number of plans/questions in bucket; (left) new test persons, (right) re-invited test persons.

The most interesting observation here is that the number of questions per time slot is

similar to the number of plans. This indicates that test persons typically ask one question per sample plan. However, they often first request a few plans before starting to ask questions. In Parent's Afternoon re-invited test person, tend to skip this initial exploration phase and directly ask questions.

A deeper examination of question-asking behavior reveals, that when no sample plan exists, i. e. the enforced goals G_i^{enf} are not solvable, users tend to request explanations more often. This is understandable, as in such cases, the existence of conflicts is evident, and conflict explanations allow test persons to directly identify why. On average, test persons ask at least one question in $53\%/51\%/51\%$ of cases when a sample plan is available and in $68\%/71\%/57\%$ of cases when there is none in Transport, Parent's Afternoon, and Rovers, respectively.

QUESTIONNAIRE EVALUATION

We now analyze the responses to the user questionnaire. Table 9 lists the Likert scale questions we asked the test persons after using the iterative planning tool. The first three questions were given to both groups, while the remaining questions addressing the helpfulness of the explanations exclusively to Q+.

			Likert scale labels	
		Question	(1)	(7)
Q− and Q+	Q1	How difficult was the task for you?	very easy	very difficult
	Q2	How satisfied are you with your achieved result.	not satisfied	very satisfied
	Q3	How confident are you, that you know which plans are possible/which goals can be achieved together?	not confident	very confident
only Q+	Q4	The possibility to ask questions helped me.	don't help at all	very helpful
	Q5	The possibility to ask questions reduced the level of difficulty.	not at all	much easier
	Q6	The questions helped me to find better plans.	don't help at all	very helpful
	Q7	The questions helped me, when I wanted to improve a plan.	don't help at all	very helpful
	Q8	The questions helped me, when the selections of goals was unsolvable.	don't help at all	very helpful
	Q9	The questions helped me to understand which plans are possible/which goals can be achieved together.	don't help at all	very helpful

Table 9: Questionnaire: Likert scale questions asked the test persons after using the iterative planning tool.

First, we compare the results for the first three questions between the domains and user groups in Figure 34. Test persons rated the task as fairly difficult, $5/6$ out of 7, with little difference between Q+ and Q−. This does not mean that the questions have no influence on the level of difficulty. It is important to remember that the people in each group, Q+ and Q−, were not aware of the alternative set-up. The result reflects the subjective difficulty of the individual task, and not the difference between the ability to ask or not to ask questions. The assessment of the test persons' confidence with regard to existing conflicts is inconclusive. Subjective user satisfaction is, however, higher for Q+ users, a trend that is again for Rovers

more evident among re-invited users.

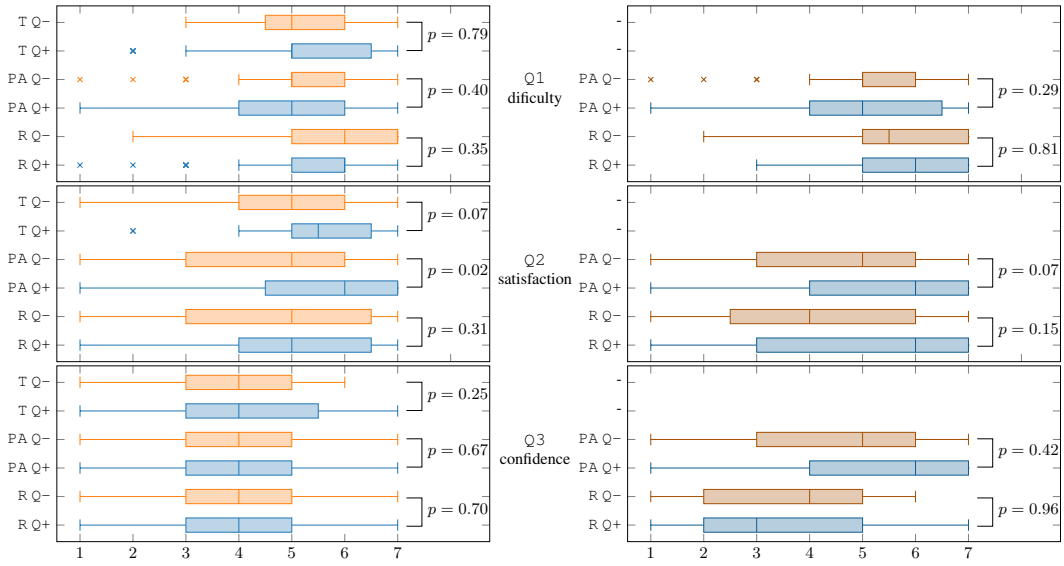


Figure 34: Questionnaire results of questions asked both groups. (left) all test person; (right) only re-invited test persons. Abbreviations: Transport `T`, Parent's Afternoon `PA`, Rovers `R`.

In the second group of questions, which addressed the helpfulness of the explanations, the responses to all questions were quite consistent. Figure 35 presents the data, which indicates an average rating of 6 out of 7 for the helpfulness of the explanations across all domains. The distribution of answers is uniform across domains, suggesting that the explanations were perceived as equally useful in each domain, despite the noted differences. Additionally, the results suggest that the explanations had a positive impact on the perceived difficulty level. If we look only at re-invited users, their subjective rating of usefulness is slightly higher compared to all users, suggesting that the explanations are more appreciated with more experience.

Qualitative Analysis We now move on to the free-text questions. The test persons were asked to:

(a) describe their problem-solving strategy (`Q+` and `Q−`)

(b) criticize the explanation facility (`Q+`)

(c) suggest explanations they would like to have (`Q−`)

We received 360 meaningful answers to (a), 152 to (b), and 161 to (c). The answers were quite diverse. In what follows, we summarize themes that were present in at least 5% of the answers and that we found to provide interesting avenues for future research. Many answers to (c) were criticisms of the iterative planning tool environment, so we group our summary into problem-solving strategies, criticisms, and desired explanations.

(a) **Problem-solving strategies** A common problem-solving strategy in both groups (19% of answers) was to start with high-utility soft goals. `Q+` (13% of answers) used the

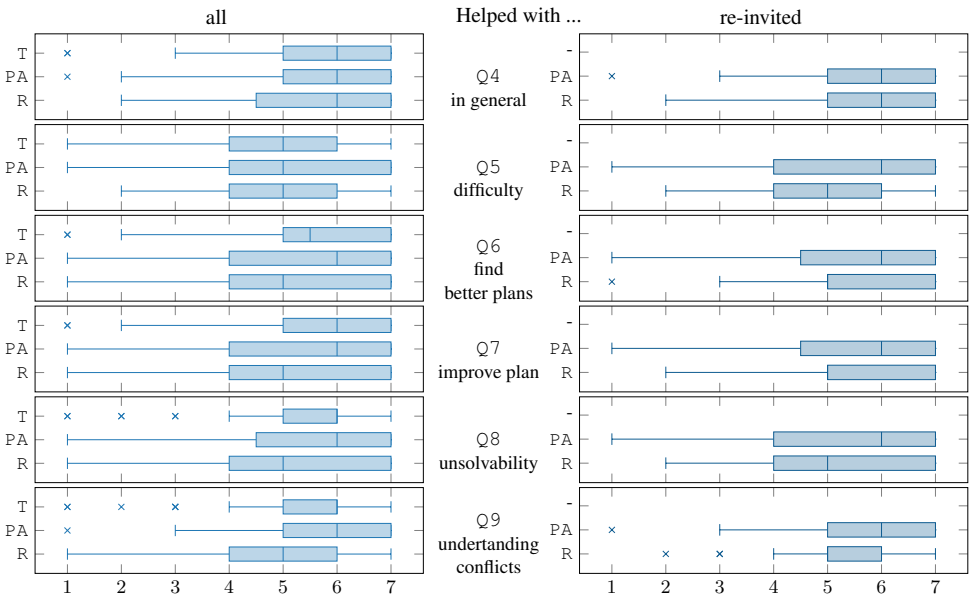


Figure 35: Questionnaire results of questions about helpfulness of explanations. (left) all test person; (right) only re-invited test persons. Abbreviations: Transport `T`, Parent's Afternoon `PA`, Rovers `R`.

question-asking interface to better understand goal conflicts: users said they used the explanations to "help guide in what to change", "to see which goals did not work together", and to help deduce "the most frequently conflicting goals".

(b) **Criticisms** The criticisms were extremely diverse. The most common theme here was visualization (5% of answers). Users asked for the ability to "draw" plans, plan animation, visualization of plans and plan-utility, for example to see how the "timeline is occupied as I go and select tasks, as well as to see the most time-efficient tasks". Users said they would like to see color-highlights overlaid on the task image, in order to "get a visual representation of what is conflicting", and to make explanations more "readable".

(c) **Explanations `Q-` users would like to have** In the explanations suggested by `Q-` users, one common theme (16%) were goal conflicts. This indicates that test persons find this form of explanation natural.

Interestingly, another common theme (14%) pertained to planning-model explanations. Users asked whether a package can "be left somewhere that another truck could pick it up", whether a truck can "go back the same way it came", or whether a guitar can be left at the "music lesson".

4.6 Discussion

In the following, we summarize the contributions and results presented in Chapter 4, discuss related work specific to this chapter and offer an outlook on future work.

SUMMARY

Conflicting objectives naturally arise during oversubscription planning (OSP). Not all user preferences are given from the start and are often not easily expressed. Therefore, we introduced an iterative planning process with explanations to help users refine their preferences, understand the conflicts between them and find a satisfactory compromise.

We proposed *conflict explanations*, contrastive explanations based on goal conflicts. Contrastive explanations, address questions of the form "Why not Q?" by demonstrating the consequences of implementing Q, the alternative proposed by the user. In our framework, Q is a set of goals not satisfied by the current sample plan π. Conflict explanations communicate to the user the goals satisfied by π they must forego in order to satisfy Q.

To provide such explanations, we compute all minimal unsolvable goal subsets (MUGS) of an OSP task. We considered an exploration of the goal lattice (GLS), in which we use the ability of symbolic planning to compactly represent the reachable state space to significantly improve the solvability check. In addition, we developed a branch-and-bound (GSBnB) approach that computes all maximal solvable goal subsets in an exhaustive state space exploration. In this context, we introduced the *new-goal-subset heuristic*, which estimates the cost to a state satisfying a goal subset that has not yet been covered. This heuristic is used to prune stats from which the maximal solvable goal subset cannot be improved. We implemented such a heuristic based on admissible estimates of singleton goal facts.

Our experiment results demonstrated the feasibility of MUGS computation for planning tasks that are sufficiently large and complex to necessitate explanations. A comparison with the most closely related, simpler problem (OSP for cardinal optimal soft goal subsets) showed comparable performance.

We expanded the framework to include temporal goals, enabling users to incorporate more complex preferences in the conflict analysis. We build on existing work on compilation of temporal goals to support LTL_f goals and introduced Action-Subset goals, as a subset of LTL_f goals with a more efficient compilation. Our results demonstrate the framework's capacity to handle a reasonable number of temporal goals.

Finally, we evaluated our conflict explanations in an online user study. The results indicate that the explanations tend to enable users to find better trade-offs, which leads us to conclude that our explanations are helpful.

RELATED WORK

In this section, we discuss further related work in terms of related approaches in XAIP, the use of minimal unsolvable sets in other areas, and the relationships between algorithms used to compute those sets.

We address contrastive explanations [Miller, 2019, 2021]. This means a user proposes an alternative Q, and seeks to know why Q is not satisfied. Approaches addressing this question, differ in the specific questions the user can ask and in how the contrast between satisfying Q and not satisfying Q is provided. The most closely related approach is by Krarup et al. [2021]. They likewise provide contrastive explanations in an iterative planning process. The question of "Why does the sample plan satisfy P rather than Q?" is addressed

by generating a plan π' that satisfies Q but not P. The explanation is then based on a comparison between π and π'. The disadvantage of using a *single* alternative π' is that there might be differences between π and π' that are not common to all plans satisfying Q. We provide an analysis of common properties of *all* alternative plans. Lindsay and Petrick [2021a] address a special case of dependency analysis based on *plan options*, focusing on plan options based on object transition sequences, learned from a set of sample plans. In addition to using the learned plan options in the framework of Krarup et al. [2021], they also propose to explain why objects use a certain transition sequence, by learning a first-order logic formula distinguishing positive and negative examples. While not addressing specific user questions Kim et al. [2019] provide contrastive explanations by learning temporal formulas distinguishing sets of sample traces. Thus, they do not explain why a particular plan does not satisfy Q by providing a contrastive plan or its properties, but they provide additional distinguishing properties of positive and negative samples for Q.

In constraint satisfaction, *Minimally Unsatisfiable Subsets* (MUSes) [Marques-Silva et al., 2013] are used analogous to MUGS. Junker [2004] employs MUSes to elucidate over-constraint problems. They assume that there exists a predefined ranking over constraints. This allows them to provide preferred explanations, instead of an explorative iterative process. To derive a satisfiable set of constraints, O'Sullivan et al. [2007] introduce an iterative process supported by explanations based on correction sets, minimal hitting sets of MUSes. In our framework, this process is comparable to starting with all soft goals and then iteratively foregoing soft goals and asking "Why is there no plan?" in each iteration. However, instead of providing all relevant MUGS O'Sullivan et al. [2007] provide a representative selection of correction sets. Gupta et al. [2022a] provide contrastive explanations based on MUSes. They leverage the MUSes in the over-constraint problem, caused by the user's alternative Q, to elucidate the necessary compromises for incorporating Q. This approach resembles one step in our iterative process, where the user wants to include soft goals Q, but to do so they have to decide which of the currently satisfied goals in $A_C(Q)$ they want to forego. MUSes are not only employed to explain unsatisfiability, but also to provide user-friendly explanations to solve a CSP [Bogaerts et al., 2020]. An example of a practical application of MUSes is a Sudoku assistant [Guns et al., 2023].

There are several algorithms for computing MUSes using different approaches. Two common approaches are iterative insertion or deletion of constraints [De Kleer and Williams, 1987, Marques-Silva, 2010], similar to our systematic strengthening and weakening goal lattice search. The objective is either MUS extraction [Marques-Silva and Lynce, 2011, Marques-Silva et al., 2013, Junker, 2001], i. e. computing one MUS, or enumerating all of them [Bailey and Stuckey, 2005, Previti and Marques-Silva, 2013, Liffiton and Sakallah, 2008, Liffiton et al., 2016]. Computing preferred MUSes involves considering a utility or ranking of constraints [Marques-Silva and Previti, 2014, Gamba et al., 2023]. For a more detailed overview of algorithm approaches and applications, we refer to [Marques-Silva, 2010].

We consider here the enumeration of all MUGS. Two state-of-the art approaches for enumerating all MUSes are CAMUS [Liffiton and Sakallah, 2008] and MARCO [Liffiton et al.,

2016]. CAMUS performs better in cases where enumerating all MUSes is tractable, while MARCO can provide some MUSes, even when enumerating all is not feasible [Liffiton et al., 2016]. CAMUS [Liffiton and Sakallah, 2008] operates in two phases. First, they calculate the maximum satisfying subsets (MSSes) using an incremental solver and iteratively adding constraints until the set of selected constraints is no longer satisfiable. Then, the hitting set relation between MUSes and MSSes is used to compute the MUSes. MARCO [Liffiton et al., 2016] on the other hand, computes one MUS/MSS at a time. Starting from a seed subset of constraints $\mathcal{C}_s$, if $\mathcal{C}_s$ is unsatisfiable, they weaken it to a MUS, and if it is satisfiable they strengthen it to a MSS. Instead of committing to one direction of exploration, they pursue a dual approach. Favoring large subsets as seeds, allows them to favor the generation of MUSes over MSSes. Our objective is to identify all MUGS, and given that planning is PSPACE-hard, we are focusing on the individual solvability test, rather than on optimized any-time goal-lattice exploration.

FUTURE WORK

There are different directions for future work, focusing on various aspects of the explanation process.

Expansion to More Expressive Planning Formalism Explanations based on MUGS could also be useful in more expressive planning formalisms, such as numeric, temporal and probabilistic planning. Each of these brings its own challenges with respect to the definition of MUGS and the computation of them. In numeric planning, the extension of MUGS to numeric facts would allow for a more natural definition of preferences such as “less than 5 units of fuel”. For our approach GSBnB, it is necessary to exhaust the search space needs, which makes heuristics proving reachability crucial to its performance. Brandao et al. [2022b] saw this as a motivation for their extension of Merge & Shrink heuristic to temporal planning. In probabilistic planning, Steinmetz et al. [2024] presented algorithms for computing conflicts between policy properties considering the uncertainty of action applications. These conflicts are determined by lower bounds on the reachability probabilities of properties, and they extend to trade-offs between different cost functions.

Visualization One point of criticism in the user study was the presentation of the explanations. The answers where presented as simple lists of the natural language description of each goal. However, this is not suitable for a large number of conflicts. To create a better interface to the explanations, a visualization would be useful. Ideally, such a visualization should allow the user to get a better overview of which goals are involved in many conflicts, but also provide better access to the conflicts with specific goals. While the former could help to identify goals that represent a bottleneck, the latter could address a specific user question. Senthooran et al. [2023] give some examples of visualizing MUCSes using simple lists and graphs grouping and connecting constraints and the MUCSes they are contained in.

Correction Sets O’Sullivan et al. [2007] use minimal correction sets, i. e. minimal sets of constraints that must be omitted to leave a satisfactory set of constraints (hitting sets

of MUGS) as explanations. In our framework, the conflict explanation $A_C(Q)$ contains all minimal conflicts that must be resolved to make Q feasible. This means, that the user has to choose one goal from each conflict in $A_C(Q)$ to forego. Correction sets represent all such minimal choices. The number of exclusion sets can be exponentially smaller and could provide a more compact answer. While conflicts primarily address the question "Why is the task not solvable?", corrections address more directly the question "How can the task be made solvable?". Thus, depending on the user's specific question, either conflicts or corrections can therefore provide a more direct or a more compact answer. With the help of conflict and correction sets Fouilhé et al. [2025] have begun to expand the questions addressed by the framework presented, incorporating frequently asked questions from end users.

Summarized Explanations The number of MUGS can be exponential in the number of soft goals. Especially when there are many similar soft goals, leading to many symmetric conflicts, it may be useful to summarize conflicts based on common features of soft Goals. This could be a simple quantification like "Because no rover can visit location L_1 and L_2." or "Because all X-ray images are in conflict with the soil sample", but also more complex summaries considering e. g. the temporal aspects of soft goals such as "First visiting the crater makes taking any soil sample infeasible." However, which features are suitable for such summaries can depend on many parameters, the current plan and the user question, but also on the question and answer history.

Large language models (LLMs) have the potential to facilitate summarizing explanations, either by explanations already available in natural language or by employing a one-step process of translating and summarizing logic-based explanations. A basic framework has been proposed that utilizes multi-agent LLMs to translate the question and then translate and summarize the explanation [Fouilhé et al., 2025].

Preferred Explanations We assume that the user preferences have not yet fully formed, may change during the iterative planning process, and are not easily expressed. Therefore, we do not include a ranking for the soft goals to filter the MUGS, to provide so-called relevant or preferred explanations [Sreedharan et al., 2018b, Vasileiou et al., 2019]. However, there may still be conflicts that are more relevant to the user's question than others and could be prioritized if there are too many conflicts to handle them all at once. A possible correlation between soft goals could be the objects or predicates involved, or for more complex temporal goals, their structure. For example, if the user asks "Why can I not take the x-ray image of the crater?", then conflicts with soft goals that consider other X-ray images or other data points of the crater might be more relevant than a conflict with using a certain connection on the other side of the map.

Explanation Approximation For tasks where the exact computation of all MUGS is not possible, an approximation of the conflicts could still provide useful information to the user. Several approaches are possible. One possibility is to generate not all, but only a subset of the MUGS, $\widetilde{\mathcal{C}} \subsetneq \mathcal{G}^{\mathsf{MUGS}}(\tau)$. This would be an under-approximation, i. e. there might be

conflicts that are not covered by the subset of MUGS. This could be done, for example, using an approach similar to Liffiton et al. [2016].

An alternative approximation, would be to cover all conflicts, but not necessarily by MUGS, but only unsolvable goal subsets, $\widetilde{\mathcal{C}}$ such that for all $C \in \mathcal{G}^{\text{MUGS}}(\tau)$ their exists a $U \in \widetilde{\mathcal{C}}$ such that $C \subseteq U$. This would be an over-approximation, i. e. the conflicts might not be minimal, and thus the conflict explanation might include goals that should be removed, although there actually exists a plan that satisfies them. To compute such an approximation, one can use an anytime approach of the branch-and-bound algorithm GSBnB, by limiting the number of expansions or the time used to explore the state space. At any time, the collected maximal solvable goal subsets (MSGS) are an under-approximation of the actual MSGS. Thus, the hitting set relation provides an over-approximation of the MUGS. How to most efficiently use the limited resources to explore the most promising and relevant parts of the search space is an open question. An approach by Eifler et al. [2024] explores the use of learned information by restricting the explored search space to a radius around a domain-dependent action policy.

CHAPTER 5

EXPLAINING GOAL CONFLICTS THEMSELVES

Explanations based on goal conflicts offer insights into the dependencies between goals, such as the conflict between taking the crater image and collecting the soil sample. However, they do not provide any information on the underlying causes of such conflicts. Exploring the causes of conflicts and potential resolutions is crucial to identifying satisfactory and understandable trade-offs.

Here, we examine the causes of soft-goal conflicts in tasks with constraints, such as resource and time window constraints. In this context, soft-goal conflicts can naturally be explained by identifying the minimal constraint relaxations under which the conflict disappears. This approach can be used to address questions such as "Why can you not take the crater image and the soil sample?" with answers like "Because one of the rovers would need 3 additional energy units." A similar approach is employed in constraint programming [Lauffer and Topcu, 2019, Senthooran et al., 2021], wherein soft constraints are introduced to suggest modifications to an unfeasible subset of constraints, thereby rendering it feasible.

These follow-up questions regarding the reason for a conflict align seamlessly with the iterative planning process. If the user asks "Why is Q not satisfied?" the answer "Because then you have to forgo $\mathcal{A}_C(Q)$" identifies the underlying conflicts. The *relaxation explanation* provides the user with the necessary information to understand why the conflicts exist and how they can be resolved. Based on this additional information, the user then has a second option to find a trade-off. The first option involves removing enforced soft goals that are in conflict with the soft goals they would like to be satisfied. The second option is to relax a constraint to resolve the conflict to allow for a plan satisfying all soft goals in conflict. For example, they might decide that it is crucial to take the crater image and the soil sample, and therefore it is worth allocating 3 extra energy units.

In the following, we define explanations of goal conflicts based on relaxations. These explanations are based on the MUGS for a given set of relaxations, which allows determining for which relaxations a certain subset of soft goals is no longer in conflict. The straightforward extension to compute the MUGS for each relaxation is to iteratively call one of the previously introduced algorithms. We present two algorithms that improve over this baseline by

exploiting the fact that information like reachable goal subsets and states can be propagated from one relaxed task to another if the latter is more relaxed. The first algorithm, MSGS propagation (GSBNBp), iteratively computes the MUGS for each increasingly relaxed planning task and propagates the MSGS to more relaxed tasks. This approach provides a growing set of reachable soft-goal subsets, which can be used to efficiently prune parts of the search space that do not contain any soft-goal subsets that have not yet been achieved. The second algorithm, Iterative Search Space Extension (ISSE), reduces redundancy by iteratively increasing one search space instead of generating a new one for each relaxed task. This is achieved by storing the search frontier for each task and using it as the starting point for more relaxed tasks, ensuring that only the newly reachable states are generated for each relaxed task. We demonstrate that these algorithms can in theory exponentially outperform the baseline, with respect to the number of generated states.

An alternative interpretation of constraint relaxations is the interpretation as soft goals rather than as task modifications. For example, by introducing soft-goals reflecting the consumed resources, the MUGS containing these *resource soft goals* allow us to determine the resource constraint relaxations needed to resolve the corresponding conflicts. This approach eliminates the need for a specialized algorithm, and the MUGS for each relaxation can be determined in a post-processing step.

We provide an empirical evaluation for resource and time window constraint domains of the two new algorithms as well, as a comparison to the compilation approach.

Papers and Contributions This chapter is based on the paper:

Rebecca Eifler, Jeremy Frank and Jörg Hoffmann
Explaining Soft-Goal Conflicts through Constraint Relaxations.
Proceedings of the 31th International Joint Conference on Artificial Intelligence (2022)

The paper was principally developed by the author in collaboration with Jeremy Frank and Jörg Hoffmann. The algorithms were developed by the author, and the implementation and experimental evaluation were also the author's work. Section 5.3 is not covered by this publication and new work of the author.

5.1 EXPLANATIONS

Let τ be an OSP task and let us consider iteration step $\delta = (G^{\text{enf}}, \pi)$. We address the scenario where the user asked questions Q and received an answer $\mathcal{A}_C(Q)$. They are now aware of the conflicts $\mathcal{C}$ between the soft goals in Q and the soft goals satisfied by the sample plan $G^{\text{true}}(\pi)$. They then pose a follow-up question for conflict $Q' \in \mathcal{C}$: "Why are Q' in conflict?" To address this question with the minimal relaxation required to resolve conflict Q', it is necessary to first establish a set of potential relaxations.

5.1.1 Task Relaxation

In general, relaxing a planning task means, allowing for more applicable action sequences. Our approach is independent of the concrete implementation of relaxation. We require the relaxation τ' of a planning task τ to preserve all plans of τ, the action cost, and the satisfiability of soft goal subsets.

Definition 30: Relaxed Planning Task

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be a OSP task and $\Pi(\tau)$ the set of all plans for task τ. Then $\tau' = (V', A', c', I', G^{\text{hard}}, G^{\text{soft}}, b')$ is a relaxed task of τ (denoted by $\tau \sqsubseteq \tau'$) iff $\Pi(\tau) \subseteq \Pi(\tau')$ and for all $\pi \in \Pi(\tau)$ we have $I[\![\pi]\!] \cap G^{\text{soft}} = I'[\![\pi]\!] \cap G^{\text{soft}}$, for all $a \in A \cap A' : c(a) = c(a')$ and $b \leq b'$.

By relaxing a task, larger soft goal subsets can be rendered satisfiable and thus conflicts can be resolved.

Proposition 12: Monotonicity MSGS

Let τ and τ' be OSP tasks. If $\tau \sqsubseteq \tau'$ then for all $G \in \mathcal{G}^{\text{MSGS}}(\tau)$ there exists a set G' in $\mathcal{G}^{\text{MSGS}}(\tau')$ such that $G \subseteq G'$.

Proof:
Let $G \in \mathcal{G}^{\text{MSGS}}(\tau)$. Thus, there exists a plan $\pi \in \Pi(\tau)$ such that $G \subseteq I[[\pi]]$. Since $\tau \sqsubseteq \tau'$ we have $\Pi(\tau) \subseteq \Pi(\tau')$ and thus $\pi \in \Pi(\tau')$. From $I[\![\pi]\!] \cap G^{\text{soft}} = I'[\![\pi]\!] \cap G^{\text{soft}}$ follows that there exists $G' \in \mathcal{G}^{\text{MSGS}}(\tau')$ such that $G \subseteq G'$.

□

Both MSGS and MUGS increase in size when a task is relaxed. Larger MSGS indicate the satisfaction of more soft goals, while larger MUGS indicate weaker conflicts, since more soft goals are required to lead to unsolvability. However, it should be noted that all goals could ultimately be solvable and then no MUGS exists.

Proposition 13: Monotonicity MUGS

Let τ and τ' be OSP tasks with goals G^{hard} and G^{soft} such that $G^{\text{hard}} \cup G^{\text{soft}}$ is not solvable in τ'. If $\tau \sqsubseteq \tau'$ then for all $G \in \mathcal{G}^{\text{MUGS}}(\tau)$ there exists a set G' in $\mathcal{G}^{\text{MUGS}}(\tau')$ such that $G \subseteq G'$.

Proof:
Because $G^{\text{hard}} \cup G^{\text{soft}}$ is not solvable in τ', $\mathcal{G}^{\text{MUGS}}(\tau')$ is not empty. Let $G \in \mathcal{G}^{\text{MUGS}}(\tau)$, which means that for all $G' \subsetneq G$ there is a plan $\pi \in \Pi(\tau)$ such that $G \subseteq I[\![\pi]\!]$. Since $\tau \sqsubseteq \tau'$ we have $\Pi(\tau) \subseteq \Pi(\tau')$ and for all $\pi \in \Pi(\tau)$ it holds that $I[\![\pi]\!] \cap G^{\text{soft}} = I'[\![\pi]\!] \cap G^{\text{soft}}$.

Therefore, either $G \in \mathcal{G}^{\text{MUGS}}(\tau')$ or G is solvable in τ' and therefore there exists a $G'' \in \mathcal{G}^{\text{MUGS}}(\tau')$ such that $G \subseteq G''$ because $\mathcal{G}^{\text{MUGS}}(\tau')$ is not empty.

□

To provide explanations, we rely on a given set of relaxed tasks. Relaxations or abstractions are a common way to compute heuristics. For example one can, modify action semantics to accumulate variable values instead of updating them, called delete relaxation [Bonet and Geffner, 2001]. Alternatively, one can abstract the state space by grouping concrete states into abstract states. The abstraction function can be based on the projection of individual variables [Culberson and Schaeffer, 1998, Edelkamp, 2001], the abstraction of individual variable domains [Domshlak et al., 2009], the cross product of subsets of domains [Seipp and Helmert, 2013] or any arbitrary mapping [Helmert et al., 2007].

While these relaxations can lead to strong heuristics, they are most likely of limited use for explanations. Since the relaxations offered as explanations should not only inform the user of the reasons behind a conflict, but also how it can be resolved, the relaxation should reflect a modification of the planning task that can be realized in the real-world problem. For example, the projection of whole variables might not be suitable. In addition, the relaxations should not be too coarse, i. e. relaxing the task too much and in too large steps. Ideally, they reflect a gradual relaxation of the task with steps that affect the conflicts. Thus, identifying meaningful and actionable relaxations is not trivial. We operationalize our framework with relaxations based on resource and time window constraints, whose concrete bounds are specified by the user. We leave the automatic identification of suitable relaxations for future work.

A set of relaxation which these properties, serves as the foundation to define relaxation explanations for goal conflicts.

5.1.2 Explanations Based on Task Relaxations

Given the question "Why are Q in conflict?" where $Q \in \mathcal{G}^{\text{MUGS}}(\tau)$ and a set of relaxed tasks $\mathbb{T}$ for τ, we use tasks $T \subseteq \mathbb{T}$ for which Q is no longer a conflict as explanation. Ideally, T should consist of the *minimally* relaxed tasks that resolve the conflict. Technically, any relaxed task that resolves the conflict could serve as an explanation. However, using the minimal relaxed task, leads to the most informative explanations based on the given set of relaxed tasks. It provides the tightest bounds, for example, the least amount of fuel that needs to be added.

Definition 31: Minimally Relaxed Task

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, $\mathbb{T}$ a set of relaxed tasks τ and $C \in \mathcal{G}^{\text{MUGS}}(\tau)$ a conflict, $\tau' \in \mathbb{T}$ is **minimally relaxed**, denoted by $\text{relax}_{min}(\mathbb{T}, C, \tau')$, if $C \notin \mathcal{G}^{\text{MUGS}}(\tau')$ and for all $\tau'' \in \mathbb{T} : \tau'' \sqsubseteq \tau' \wedge \tau'' \neq \tau' \rightarrow C \in \mathcal{G}^{\text{MUGS}}(\tau'')$.

A minimally relaxed task for conflict C is a task in which C is not a conflict, but for all less relaxed tasks, it is. The relaxation explanation for a conflict can then be defined as follows:

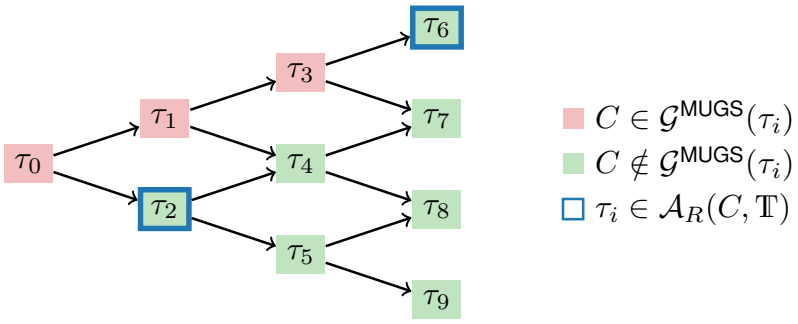
DEFINITION 32: RELAXATION EXPLANATION

Let τ be an OSP task and $\mathbb{T}$ a set of relaxed tasks τ.
$(Q, \mathcal{A}_R(Q, \mathbb{T}))$ is a **relaxation explanation** where $Q \in \mathcal{G}^{\text{MUGS}}(\tau)$ is the user question $Q \in \mathcal{G}^{\text{MUGS}}(\tau)$ and $\mathcal{A}_R(Q, \mathbb{T}) := \{\tau' \in \mathbb{T} \mid \text{relax}_{min}(\mathbb{T}, Q, \tau')\}$ is the answer to question Q.

The natural language formulation of the question-answer pair is "Why are the soft goals in Q in conflict?" — "Because to resolve conflict Q, task τ would need to be relaxed at least to one relaxation in $\mathcal{A}_R(Q, \mathcal{T})$". A generic example is provided in Example 12 for more concrete examples we refer to Section 5.4 and Example 22.

EXAMPLE 12: RELAXATION EXPLANATION

To illustrate the explanation for conflict C we use the diagram below. The minimal relaxed tasks for C and the therefore given explanation is $\mathcal{A}_R(C, \mathbb{T}) = \{\tau_2, \tau_6\}$. The explanation in natural language is as follows: "To resolve conflict C, task τ would need to be relaxed at least to τ_2 or τ_6". τ_4 is not part of the explanations because $\tau_2 \sqsubseteq \tau_4$, which means τ_4 is not a minimally relaxed task.



Hasse diagram for a sample set of relaxations. $\tau_i \to \tau_j$ means $\tau_i \sqsubseteq \tau_j$.

A few comments are necessary from a conceptual and technical point of view:

- Whether the iteration step δ is solvable or not, does not affect the explanation. For both cases, the follow-up question is "Why are the soft goals in Q in conflict?". If the iteration step is solvable then Q refers to one of the conflicts revealed by the conflict explanation $\mathcal{A}_C(Q')$ of the preceding question "Why is Q' not satisfied?", while for an unsolvable iteration step Q is one of the conflicts in the answer $\mathcal{A}^{\not\iota}$ to the question "Why is there no plan for G_i^{enf}?".
- The question Q is restricted to minimal conflicts, i. e. MUGS. This is because we assume that if Q is not minimal, the question "Why are the soft goals in Q in conflict?" is first answered with the MUGS contained in Q. For more details see next section.
- The question Q is limited to one conflict. In case the user is interested in a relaxation to resolve a set of conflicts $\mathcal{C}$, they can ask one question $Q_i \in \mathcal{C}$ per conflict. The answer

resolving all conflicts are the least relaxed tasks from $\mathbb{T}$ that are at least as relaxed as the tasks in the individual answers: $\tau' \in \mathbb{T}$ for which for all $\tau'' \in \bigcup_{Q_i \in \mathcal{C}} \mathcal{A}_R(Q_i, \mathbb{T})$ holds $\tau'' \sqsubseteq \tau'$ and not exists $\tau''' \in \mathbb{T}$ such that $\tau''' \sqsubseteq \tau$.

- Given that $\mathbb{T}$ is only partially ordered, $\mathcal{A}_R(Q, \mathbb{T})$ may contain multiple relaxations. We assume that the user has not yet fully formed their preferences, including potential relaxations. Therefore, we present all minimal relaxations to the user. The user must decide, as part of the iterative planning process, whether the resolution of the conflict is worth applying one of the relaxations.

As in the previous chapter, we use the MUGS for each task to represent all dominant goal conflicts per task. Our approach requires computing the MUGS for all given relaxations.

DEFINITION 33: ALLRELAXMUGS

Let τ be an OSP task and $\mathbb{T}$ a set of relaxed tasks for τ. By **AllRelaxMUGS** we denote the algorithmic problem of computing $\mathcal{G}^{\text{MUGS}}(\tau')$ for all $\tau' \in \mathbb{T}$.

These can be computed offline, prior to the iterative planning process, to ensure responsiveness when interacting with the user.

5.1.3 INTEGRATION INTO ITERATIVE PLANNING PROCESS

The relaxation explanation expects as question a minimal conflict, i. e. one MUGS. However, the response to a conflict explanation (Definition 15) does not necessarily provide minimal conflicts.

"**Why is Q not satisfied by π?**":

Case 1: The answer $\mathcal{A}_C(Q) = \{\emptyset\}$ indicates that achieving the question Q itself is not possible. However, Q could be non-minimal or even contain several conflicts.

Case 2: The answer $\mathcal{A}_C(Q) = \{A_0, \cdots, A_n\}$ indicates that at least one soft goal in each A_i must be foregone to satisfy Q. However, it does not reveal which part of Q is in conflict with each A_i. $Q \cup A_i$ does not need to be minimal and may also contain several conflicts.

Case 3: If question Q is not in conflict with any soft goals satisfied by the sample plan, then the answer is empty, $\mathcal{A}_C(Q) = \emptyset$. Since there is no conflict preventing the user from enforcing Q in addition to the already enforced soft goals, no follow-up question is necessary.

"**Why is there no plan?**": In case there is no plan, as the enforced goals G^{enf} are not solvable, the answer $\mathcal{A}\not$ provides the minimal conflicts in G^{enf}.

For Case 1 and Case 2 the user may want to know the minimal conflicts causing the unsolvability of Q and $Q \cup A_i$ respectively. This provides the user with more fine-grained information about the conflicts in their questions or with respect to the question and the soft

goals satisfied by the plan. It also provides the minimal conflicts necessary to ask for a relaxation explanation.

DEFINITION 34: MINIMAL CONFLICT EXPLANATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task. $(Q, \mathcal{A}_M(Q))$ is **minimal conflict explanation** where $Q \subseteq G^{\text{soft}}$ is a user question such that there exists a $C \in \mathcal{G}^{\text{MUGS}}(\tau)$ with $C \subseteq Q$ and $\mathcal{A}_M(Q) = \{C \mid C \in \mathcal{G}^{\text{MUGS}}(\tau), C \subseteq Q\}$ is the answer.

This is analogous to how the question "Why is there no plan for G^{enf}?" is addressed in Definition 16. In both cases, the user has a set of unsatisfiable soft goals and wants to identify the minimal conflicts that cause the unsolvability.

A concrete example of an iteration step containing all three question types is given in Example 13.

EXAMPLE 13: ITERATION STEP WITH ALL EXPLANATION TYPES

Let's again consider our running example with 8 units of energy and only soft goals. We abbreviate $x = \texttt{uploaded}$ with x. τ_e refers to the task with e units of energy for the rover.

(1) Select G^{enf}: $G^{\text{enf}} = \{\texttt{image-ice}, \texttt{image-crater3}\}$

(2) sample plan π: $G^{\text{true}}(\pi) = G^{\text{enf}}$ and $G^{\text{false}}(\pi) = DP \setminus G^{\text{enf}}$

(3) Why not Q?: $Q = \{\texttt{soil-sample-ice}, \texttt{x-ray-image-rock}\}$
"Why is the soil sample of the ice surface and the x-ray image of the rock not satisfied by π?"

(4) $\mathcal{A}_C(Q)$: $\mathcal{A}_C(Q) = \{\{\texttt{image-ice}\}\}$
"Because to collect the soil sample of the ice surface and take the x-ray image of the rock you have to forgo the image of the ice surface."

(5) Why not Q' a conflict?: $Q' = \{\texttt{soil-sample-ice}, \texttt{x-ray-image-rock}, \texttt{image-ice}\}$
"Why is it not possible to collect the soil sample and take an image of the ice surface and take an x-ray image of the rock?"

(6) $\mathcal{A}_M(Q')$: $\mathcal{A}_M(Q') = \{\{\texttt{image-ice}, \texttt{x-ray-image-rock}\}\}$
"Because it is not possible to take the image of the ice surface and the x-ray image of the rock."

(7) Why not Q'' a conflict?: $Q'' = \{\texttt{image-ice}, \texttt{x-ray-image-rock}\}$
"Why is it not possible to take the image of the ice surface and the x-ray image of the rock"

> (8) $\mathcal{A}_R(Q'')$: $\mathcal{A}_R(Q'') = \{\tau_9\}$
> "Because you need one additional unit of energy to take the image of the ice surface and the x-ray image of the rock."

For one iteration step within the iterative planning process, this results in the interaction sequence shown in Figure 36.

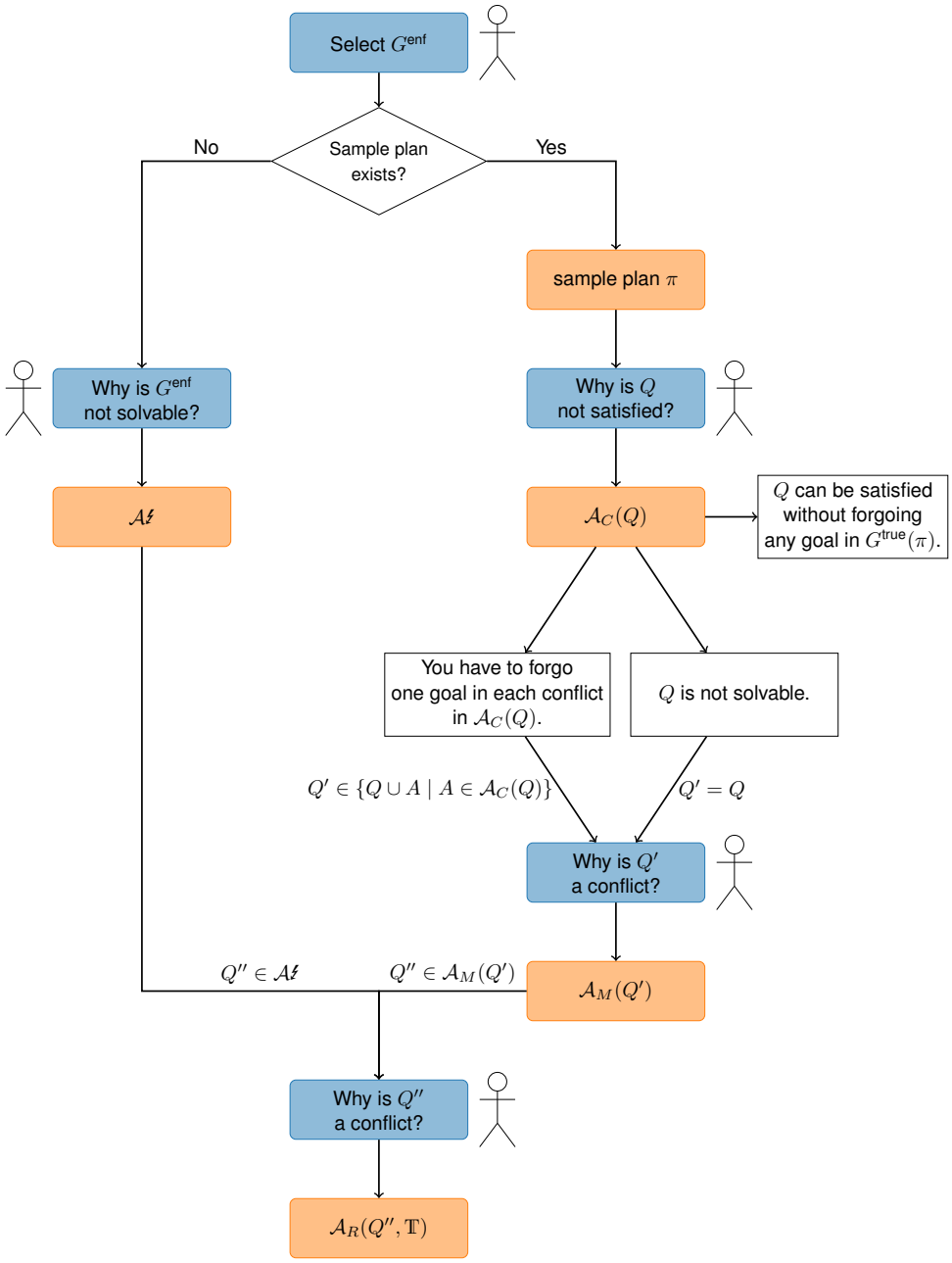


Figure 36: Interaction options of the user and questions they can ask within one iteration step.

5.2 AllRelaxMUGS Algorithms

To address any question of the form: "Why are soft goals Q in conflict?", it is necessary to compute the MUGS of all relaxed tasks $\mathbb{T}$. A straightforward approach would be to utilize any of the algorithms introduced in the previous chapter, either goal lattice search GLS (Section 4.2.1) or branch-and-bound search (GSBnB) (Section 4.2.2), to compute the MUGS for each task in $\mathbb{T}$ individually. However, this approach would entirely overlook the relationship between the tasks in $\mathbb{T}$. These tasks are relaxations of the original task τ and often relaxations of each other. Therefore, we propose the following two approaches based on GSBnB, which propagate information from less relaxed to more relaxed tasks. To do so, we need to traverse the relaxed tasks following the partial order provided by $\sqsubseteq$. This traversal is based on the exploration of the relaxation graph, defined in the next section.

Relaxation Graph For the implementation of our algorithms we assume that a *relaxation graph* with the relaxed tasks $\mathbb{T}$ as nodes and edges based on an under-approximation of $\sqsubseteq$ with the following properties is given.

Definition 35: Relaxation Graph

Let τ be a planning task and $\mathbb{T}$ as set of relaxed tasks for τ. The directed graph $\mathcal{T} = (\mathbb{T}, E)$ with vertices $\mathbb{T}$ and edges E is a **relaxation graph** for τ if

(1) $\tau \in \mathbb{T}$,

(2) for all $(\tau', \tau'') \in E$ it holds $\tau' \neq \tau''$,

(3) for all $\tau', \tau'' \in \mathbb{T}$ it holds that $(\tau', \tau'') \in E \rightarrow \tau' \sqsubseteq \tau''$,

(4) it exists $\tau^* \in \mathbb{T}$ such that for all $\tau' \in \mathbb{T}$, $\tau' \sqsubseteq \tau^*$, and

(5) for all $\tau' \in \mathbb{T} \setminus \{\tau, \tau^*\}$ there exists a path from τ via τ' to τ^*.

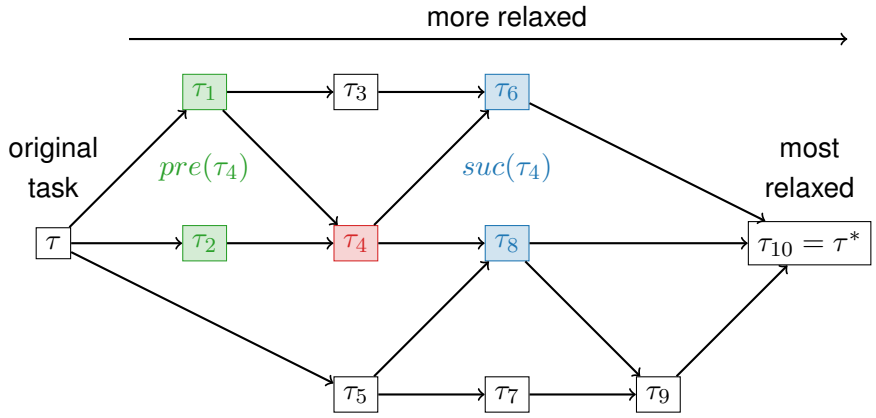
Task τ must be a vertex of the relaxation graph (1) and self-loops are not allowed (2). (3) ensures that the edges reflect an under-approximation of $\sqsubseteq$. Due to (2) and (3) τ has no incoming edges. Our algorithms require a most relaxed task (4). We refer to this task as τ^*. All tasks in $\mathbb{T}$ must be located on a path between task τ and the most relaxed task τ^* (5).

In the following algorithms, we will need the functions $pre(\tau')$ and $suc(\tau')$ denoting the predecessor (less relaxed) and successor (more relaxed) tasks of τ' within relaxation graph $\mathcal{T} = (\mathbb{T}, E)$. They are defined as $pre(\tau') = \{\tau'' \in \mathbb{T} \mid (\tau'', \tau') \in E\}$ and $suc(\tau') = \{\tau'' \in \mathbb{T} \mid (\tau', \tau'') \in E\}$. An example relaxation graph is depicted in Example 14.

Example 14: Relaxation Graph

Below the relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ with vertices $\mathbb{T} = \{\tau, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8, \tau_9, \tau_{10}\}$ and edges $E = \{(\tau, \tau_1), (\tau, \tau_2), (\tau, \tau_5), (\tau_1, \tau_3), (\tau_1, \tau_4), (\tau_3, \tau_6), (\tau_4, \tau_8), (\tau_5, \tau_8), (\tau_5, \tau_7), (\tau_6, \tau_{10}), (\tau_8, \tau_{10}), (\tau_7, \tau_9), (\tau_9, \tau_{10})\}$ is depicted.

The predecessor of τ_4 are all tasks with an outgoing connection with τ_4 and the successor task have an incoming connection with τ_4. τ_{10} is the most relaxed task.



We traverse the relaxation graph $\mathcal{T} = (\mathbb{T}, E)$, starting in τ, to compute all MUGS for each task in $\mathbb{T}$. In the following we introduce two algorithms that during this exploration propagate information from less to more relaxed tasks.

5.2.1 MSGS Propagation

As described in Section 4.2.2, GSBnB explores the state space while maintaining all subset maximal solvable goal subsets $\mathcal{M}$. To decrease the number of explored states, a heuristic estimation based on the already encountered soft goal subsets $\mathcal{M}$ is used for pruning. If no goal subset that can improve $\mathcal{M}$ can be reached within the cost bound, the state is pruned. The more soft goal subsets are reached, the more effective the pruning becomes. This process can lead to more pruning if $\mathcal{M}$ is initialized with an under-approximation of $\mathcal{G}^{\text{MSGS}}(\tau)$ (all maximal solvable goal subsets of task τ). Such an under-approximation can be provided by the MSGS of less relaxed tasks. In the following, we exploit this property to extend GSBnB to compute AllRelaxMUGS.

Given a relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ for τ, we compute the MSGS for all $\tau' \in \mathbb{T}$ by traversing $\mathcal{T}$, starting in τ, and computing the MSGS for each task individually. The MSGS are then propagated from less to more relaxed tasks.

The pseudo-code of GSBnB *with MSGS propagation* (GSBnBp) is outlined in Algorithm 6. $\mathcal{M}$ (line 3) is a map from tasks to sets of soft-goal subsets storing the MSGS for each task. The aforementioned propagation of MSGS is realized by initializing GSBnB with all MSGS reached in the predecessor tasks of τ (line 8). The heuristic functions depend on the task, thus they needs to be recomputed for each task τ. The processing of relaxed tasks is done in the order resulting from the traversal of the relaxation graph $\mathcal{T}$. A task is selected based on whether all its predecessors have been processed, and the order of incomparable tasks is resolved randomly (line 6). If a task solves all hard and soft goals, all more relaxed tasks are also solvable, and thus, they are skipped (line 10). The MSGS can then be used to compute the MUGS for each task, as done in Section 4.2.

Algorithm 6 MSGS Propagation Search (GSBNBp)

Given: relaxation graph $\mathcal{T} = (\mathbb{T}, E)$, new-goal-subset heuristic H_p, heuristic H_g
function GSBNBP($\mathcal{T}, H_p, H_g$)
  $\mathcal{M} \leftarrow \{\}$ ▷ *map of MSGSs*
  $T_p \leftarrow \{\}$ ▷ *processed tasks*
  while $|T_p| < |\mathbb{T}|$ **do**
    $\tau \leftarrow$ RANDOM($\{\tau' \in \mathbb{T} \setminus T_p | pre(\tau') \subseteq T_p\}$) ▷ *next relaxed task*
    $T_p \leftarrow T_p \cup \{\tau\}$
    $\widetilde{M} \leftarrow \max_{\subseteq}(\bigcup_{\tau' \in pre(\tau)} \mathcal{M}[\tau'])$ ▷ *propagate MSGS*
    $\mathcal{M}[\tau] \leftarrow$GSBNB($\tau, H_p(\tau), H_g(\tau), \widetilde{M}$)
    if $G^{\text{hard}} \cup G^{\text{soft}} \in \mathcal{M}[\tau]$ **then** ▷ *skip solvable tasks*
      $T_p \leftarrow T_p \cup \{\tau' \in \mathbb{T} | \tau'$ is reachable from τ in $\mathcal{T}\}$
  return $\{HIT(\overline{\mathcal{M}(\tau)}) \mid \tau \in \mathbb{T}\}$

PROPOSITION 14: SOUNDNESS AND COMPLETENESS OF GSBNBP

Given a relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ GSBNBp computes the MUGS of all tasks in $\mathbb{T}$.

Proof:
It follows from Proposition 12, that MSGS grow monotonically from less to more relaxed task. Thus all MSGS$\widetilde{M}$ reachable within the predecessor of task τ are also reachable in τ. Thus, the soundness and completeness of GSBNBp follows from the soundness and completeness of GSBNB.

□

5.2.2 ITERATIVE SEARCH SPACE EXTENSION

The MSGS propagation approach performs one exhaustive state space search for each task. However, since we consider relaxed tasks, there is no need to generate individual search spaces. Instead, one can iteratively extend the search space of less relaxed tasks with states, that are only reachable in more relaxed tasks. This way, one must not generate *corresponding states* multiple times.

DEFINITION 36: CORRESPONDING STATES

Let τ and τ' be OSP tasks, with initial states I, I', actions A, A' and states $\mathcal{S}$, $\mathcal{S}'$ respectively.
Two states $s \in \mathcal{S}$ and $s' \in \mathcal{S}'$ are **corresponding states**, iff there exists an action sequence $\pi \subseteq A \cap A'$ such that $I[\![\pi]\!] = s$ and $I'[\![\pi]\!] = s'$ holds.
The function $\kappa^{\tau}_{\tau'} : \mathcal{S} \mapsto \mathcal{S}' \cup \epsilon$ maps each state to its corresponding state or to ϵ if no such state exists. It is called **corresponding state mapping**.

States correspond to each other if they can be reached with the same action sequence. For instance, if the relaxation allows to use a second rover, all action sequences that only use the first rover are applicable in the original task and in the relaxation, resulting in corresponding states. Since relaxations preserve all plans, if a task is relaxed then each goal state of the task has a corresponding goal state in the relaxed task.

PROPOSITION 15: CORRESPONDING STATES FOR RELAXED TASKS

Let τ and τ' be OSP tasks. If $\tau \sqsubseteq \tau'$ then for all states $s \in \mathcal{S}^r(\Theta_\tau)$, there is a state $s' \in \mathcal{S}^r(\Theta_{\tau'})$ such that $\kappa_\tau^{\tau'}(s') = s$.

Proof:
Let $\pi \in \Pi(\tau)$ and $s = I[\![\pi]\!]$. Since $\tau \sqsubseteq \tau'$ we have $\Pi(\tau) \subseteq \Pi(\tau')$. Thus, there exists a state $s' \in \mathcal{S}'$ such that $I'[\![\pi]\!] = s'$ and hence $\kappa_\tau^{\tau'}(s') = s$.

□

Each goal state of τ has a corresponding goal state in the relaxed task τ', satisfying the same soft goals. Thus, we can compute $\mathcal{G}^{\text{MSGS}}(\tau)$ by restricting the state space of τ' to the states with corresponding reachable states in τ. To do so, we use the cost bound of τ and only allow transitions in the relaxed state space, that also exist between corresponding states in the non-relaxed state space.

DEFINITION 37: RESTRICTED STATE SPACE

Let τ and τ' be OSP tasks with state spaces $\Theta_\tau = (\mathcal{S}, L, c, T, I, \mathcal{S}_G, b)$ and $\Theta_{\tau'} = (\mathcal{S}', L', c', T', I', \mathcal{S}_G', b')$ and $\kappa_\tau^{\tau'}$ the corresponding state mapping between τ' and τ. The state space of τ' **restricted** to τ is the state space $\Theta_{\tau'}^{\tau} = (\mathcal{S}', L', c', T_\tau, I', \mathcal{S}_G', b)$ where $T_\tau = \{(s, a, s') \in T' \mid (\kappa_\tau^{\tau'}(s), a, \kappa_\tau^{\tau'}(s')) \in T\}$.

Then the MSGS of task τ can be computed based on the state space of a relaxed task τ', restricted to τ.

PROPOSITION 16: $\mathcal{G}^{\text{MSGS}}$ FROM RESTRICTED STATE SPACE

Let τ and τ' be OSP tasks where $\tau \sqsubseteq \tau'$. Then $\mathcal{G}^{\text{MSGS}}(\Theta_\tau) = \mathcal{G}^{\text{MSGS}}(\Theta_{\tau'}^{\tau})$.

Proof:
Using Proposition 16 we have $\mathcal{G}^{\text{MSGS}}(\Theta_\tau) = \max_\subseteq\{\cup_{s \in \mathcal{S}_G^r(\Theta_\tau)}(s \cap G^{\text{soft}})\}$ and $\mathcal{G}^{\text{MSGS}}(\Theta_{\tau'}^{\tau}) = \max_\subseteq\{\cup_{s \in \mathcal{S}_G^r(\Theta_{\tau'}^{\tau})}(s \cap G^{\text{soft}})\}$, where $\mathcal{S}_G^r(x)$, are the reachable goal states of x.

$\{G^{\text{soft}} \cap s | s \in \mathcal{S}_G^r(\Theta_\tau)\} \subseteq \{G^{\text{soft}} \cap s | s \in \mathcal{S}_G^r(\Theta_{\tau'}^{\tau})\}$: According to Proposition 15, every state $s \in \mathcal{S}_G^r(\Theta_\tau)$ has a corresponding state $s' \in \mathcal{S}_G^r(\Theta_{\tau'})$ that satisfies the same

soft goals. Since the restriction of $\Theta_{\tau'}$ to τ maintains the reachability of all states that have a corresponding state in Θ_τ, the states corresponding to $\mathcal{S}^r_G(\Theta_\tau)$ are also reachable in $\mathcal{S}^r_G(\Theta^\tau_{\tau'})$. Thus, $\{\kappa^\tau_{\tau'}(s) \mid s \in \mathcal{S}^r_G(\Theta_\tau)\} \subseteq \mathcal{S}^r_G(\Theta^\tau_{\tau'})$ and consequently $\{G^{\text{soft}} \cap s | s \in \mathcal{S}^r_G(\Theta_\tau)\} \subseteq \{G^{\text{soft}} \cap s | s \in \mathcal{S}^r_G(\Theta^\tau_{\tau'})\}$

$\{G^{\text{soft}} \cap s | s \in \mathcal{S}^r_G(\Theta_\tau)\} \supseteq \{G^{\text{soft}} \cap s | s \in \mathcal{S}^r_G(\Theta^\tau_{\tau'})\}$: We have $\{\kappa^\tau_{\tau'}(s) \mid s \in \mathcal{S}^r_G(\Theta_\tau)\} \supseteq \mathcal{S}^r_G(\Theta^\tau_{\tau'})$, because if this is not the case, there would be a state $s \in \mathcal{S}^r_G(\Theta^\tau_{\tau'})$ such that there exists a plan π with $s = I'[\![\pi]\!]$ and π is not applicable in I. However, since $\Theta^\tau_{\tau'}$ only contains transitions that also exist in Θ_τ, π must also be applicable in I. From this, and that corresponding goal states satisfy the same soft goals $\{G^{\text{soft}} \cap s | s \in \mathcal{S}^r_G(\Theta_\tau)\} \supseteq \{G^{\text{soft}} \cap s | s \in \mathcal{S}^r_G(\Theta^\tau_{\tau'})\}$ follows.

□

The objective is to compute the MSGS for all relaxed tasks, by iteratively extending the search space. Next, we describe how to transition from the states explored for τ to then compute $\mathcal{G}^{\text{MSGS}}(\tau')$. This is possible, because the reachable states in $\Theta^\tau_{\tau'}$ are a *prefix* of $\Theta_{\tau'}$.

DEFINITION 38: SEARCH SPACE PREFIX

Let τ and τ' be OSP tasks with state spaces $\Theta_\tau = (\mathcal{S}, L, c, T, I, \mathcal{S}_G)$ and $\Theta_{\tau'} = (\mathcal{S}, L, c, T', I, \mathcal{S}_G)$.
Θ_τ is called a **prefix** of $\Theta_{\tau'}$ if all states $s \in \mathcal{S}$ reachable in Θ_τ are also reachable in $\Theta_{\tau'}$, $\mathcal{S}^r(\Theta_\tau) \subseteq \mathcal{S}^r(\Theta_{\tau'})$.

PROPOSITION 17: RESTRICTED SEARCH SPACE IS PREFIX

Let τ and τ' be OSP tasks where $\tau \sqsubseteq \tau'$. Then $\Theta^\tau_{\tau'}$ is a prefix of $\Theta_{\tau'}$.

Proof:
According to Definition 37 we have $\Theta^{\tau'} = (\mathcal{S}', L', c', T', I', \mathcal{S}_G', b)$ and $\Theta^\tau_{\tau'} = (\mathcal{S}', L', c', T_\tau, I', \mathcal{S}_G', b)$. Since $T_\tau \subseteq T'$ and they have the same cost function and cost bound all states reachable in $\Theta^\tau_{\tau'}$ are also reachable in $\Theta_{\tau'}$.

□

For all states $s \in \mathcal{S}'$ that are only reachable in $\Theta_{\tau'}$, the path from I' to s can be decomposed into two parts. The first part is applicable in $\Theta^\tau_{\tau'}$ and the second part is applicable only in the non-restricted state space $\Theta_{\tau'}$. The border between these two parts are the *frontier* states. They are the first states that are not reachable in the restricted search space.

PROPOSITION 18: FRONTIER

Let τ and τ' be OSP tasks with state spaces $\Theta_\tau = (\mathcal{S}, L, c, T, I, \mathcal{S}_G)$ and $\Theta_{\tau'} = (\mathcal{S}, L, c, T', I, \mathcal{S}_G)$, such that Θ_τ is a prefix of $\Theta_{\tau'}$. For every state $s \in \mathcal{S}^r(\Theta_{\tau'}) \setminus \mathcal{S}^r(\Theta_\tau)$ (that is reachable in $\Theta_{\tau'}$ but not in Θ_τ) there exists a transition sequence $\sigma = (I, l_0, s_1) \cdots (s_{i-1}, l_{i-1}, s_i)(s_i, l_i, s_{i+1})(s_{i+1}, l_{i+1}, s_{i+2}) \cdots (s_{n-1}, l_{n-1}, s)$ from I to s such that

- $(s_k, l_k, s_{k+1}) \in T$ for $k < i$
- $(s_i, l_i, s_{i+1}) \notin T$

Proof:
Let $s \in \mathcal{S}$ be a state reachable in $\Theta_{\tau'}$ but not in Θ_τ. Let σ be a shortest path from I to s in $\Theta_{\tau'}$. Since s is not reachable in Θ_τ there must be a transition $(s, a, s') \in \sigma$ such that $(s, a, s) \notin T$. Let $\sigma = (I, l_0, s_1) \cdots (s_{i-1}, l_{i-1}, s_i)(s_i, l_i, s_{i+1})(s_{i+1}, l_{i+1}, s_{i+2}) \cdots (s_{n-1}, l_{n-1}, s)$. Select i such that for all $k < i$ $(s_k, l_k, s_{k+1}) \in T$, and $(s_i, a_i, s_{i+1}) \in T$ holds. This is always possible, because i can be 0.

□

We call s_{i+1}, the first state in $\Theta_{\tau'}$, a **frontier state** and denote the set of all **frontier states** with $F^\tau_{\tau'}$. The frontier states together with the MSGS of task τ allow to compute the MSGS for the relaxed task τ'.

THEOREM 3: $\mathcal{G}^{\text{MSGS}}$ BY EXTENTION OF RESTRICTED SEARCH SPACE

Let τ and τ' be OSP tasks where $\tau \sqsubseteq \tau'$ with cost bound b and b' respectively. Let $\Theta^\tau_{\tau'} = (\mathcal{S}', L', c', T_\tau, I', \mathcal{S}_G{}', b)$ be the state space of τ' restricted to τ with frontier states $F^\tau_{\tau'}$.
Then

$$\mathcal{G}^{\text{MSGS}}(\tau') = \max_{\subseteq}\{\mathcal{G}^{\text{MSGS}}(\Theta^\tau_{\tau'}) \cup \bigcup_{f \in F^\tau_{\tau'}} \mathcal{G}^{\text{MSGS}}((\mathcal{S}', L', c', T', f, \mathcal{S}_G{}', b' - g(f)))\}$$

where $g(f)$ is the cost of a cheapest path from I' to f in $\Theta^\tau_{\tau'}$.

Proof:
$\subseteq$: Let $G \in \mathcal{G}^{\text{MSGS}}(\tau')$. If

(a) $G \in \mathcal{G}^{\text{MSGS}}(\tau)$: If $G \in \mathcal{G}^{\text{MSGS}}(\tau)$ then $G \in \mathcal{G}^{\text{MSGS}}(\Theta^\tau_{\tau'})$ follows from Proposition 16. It remains to be shown that $\max_\subseteq$ does not remove G. If $G \in \mathcal{G}^{\text{MSGS}}(\tau')$ then there exists no $G' \supsetneq G$, for which there exists a stare $s \in \mathcal{S}^r(\Theta_{\tau'})$ such that $G' \subseteq s$. So no $G' \supsetneq G$ is added and thus G is not removed by $\max_\subseteq$.

(b) $G \notin \mathcal{G}^{\text{MSGS}}(\tau)$: This means there is a state $s \in \mathcal{S}^r(\Theta_{\tau'}) \setminus \mathcal{S}^r(\Theta^\tau_{\tau'})$ such that $G \subseteq s$.

Let $\sigma = (I', a_0, s_1) \cdots (s_{i-1}, a_{i-1}, s_i)(s_i, a_i, f)(f, a_{i+1}, s_{i+2}) \cdots (s_{n-1}, a_{n-1}, s)$ be a cheapest transition sequence reaching s in $\Theta_{\tau'}$. From Proposition 18 follows that there is a frontier state in σ. Let f be this frontier state. Thus, s is reachable in state space $(\mathcal{S}', L', c', T', f, \mathcal{S}_G{}', b' - \text{cost}(a_0 \cdots a_i))$ via the transition sequence (s_i, a_i, f) $((f, a_{i+1}, s_{i+2}) \cdots (s_{n-1}, a_{n-1}, s)$. Because G is maximal in τ' there exists no state $s' \in \mathcal{S}'$ and $G' \supsetneq G$ such that $G' \subseteq s'$. Thus, $\max_{\subseteq}$ does not remove G.

$\supseteq$:

(a) $G \in \mathcal{G}^{\text{MSGS}}((\mathcal{S}', L', c', T', f, \mathcal{S}_G{}', b' - g(f)))$ for any $f \in F^{\tau}_{\tau'}$: Let $s \in \mathcal{S}^r((\mathcal{S}', L', c', T', f, \mathcal{S}_G{}', b' - g(f)))$ such that $G \subseteq s$ and $(f, a_{i+1}, s_{i+2}) \cdots (s_{n-1}, a_{n-1}, s)$ the transition sequence to reach s from f. Because f is a frontier state (Proposition 18), there exists a transition sequence $\sigma = (I', a_0, s_1) \cdots (s_{i-1}, a_{i-1}, s_i)(s_i, a_i, f)(f, a_{i+1}, s_{i+2}) \cdots (s_{n-1}, a_{n-1}, s)$ in $\Theta_{\tau'}$ that reaches s. Thus, G is solvable in τ'.

It remains to be shown that G is subset-maximal among $\mathcal{G}^{\text{MSGS}}(\tau')$.

- If there exists a frontier state f' such that a superset $G' \supsetneq G$ is reachable from f' then $\max_{\subseteq}$ would remove G.
- Let $G \subseteq G^{\text{soft}}$ be subset-maximal among all soft goal subset reachable from any frontier state. Now, assume G is not subset-maximal in τ', i. e. there exists a state $s \in \mathcal{S}^r_G(\Theta_{\tau'})$ such that $G' = G^{\text{soft}} \cap s \supsetneq G$. If $s \in \mathcal{S}^r_G(\Theta^{\tau}_{\tau'})$, then G' is covered by (b) and G removed by $\max_{\subseteq}$. If $s \in \mathcal{S}^r(\Theta_{\tau'}) \setminus \mathcal{S}^r(\Theta^{\tau}_{\tau'})$, following the argumentation from above ($\subseteq$: (a)) there is a frontier state f such that s is reachable from f. However, this contradicts with G being subset-maximal among all frontier states.

(b) $G \in \mathcal{G}^{\text{MSGS}}(\Theta^{\tau}_{\tau'})$: $G \in \mathcal{G}^{\text{MSGS}}(\tau)$ follows from Proposition 16. It remains to be shown that G is either subset-maximal in $\mathcal{G}^{\text{MSGS}}(\tau')$ or removed by $\max_{\subseteq}$. From Proposition 12 follows, that there is a $G' \in \mathcal{G}^{\text{MSGS}}(\tau')$ such that $G \subseteq G'$. If $G' = G$ then there is no state $s' \in \mathcal{S}'$ and $G'' \supsetneq G'$ such that $G'' \subseteq s'$. Thus, $\max_{\subseteq}$ does not remove G. If $G \subsetneq G'$, then G' or a superset is contained in $\mathcal{G}^{\text{MSGS}}((\mathcal{S}', L', c', T', f, \mathcal{S}_G{}', b' - g(f)))$ because of $(\supseteq (a))$ and thus $\max_{\subseteq}$ removes G.

□

According to Theorem 3, allRelaxMUGS for a set of relaxations $\mathbb{T}$ of a given relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ can be computed by iteratively extending the search space of the most relaxed task τ^*. The initialization of the search for more relaxed tasks is based on the stored frontier nodes from predecessor tasks and their MSGS. The pseudo-code of the *Iterative Search Space Extension* (ISSE) algorithm is outlined in Algorithm 7 and 8. The algorithm is divided into two functions, ISSE (Algorithm 7), which iterates over the relaxed tasks and maintains their MSGS and search frontiers, and EXTEND (Algorithm 8), which explores the state space for one relaxed task.

Algorithm 7 Iterative Search Space Extension (ISSE)

1: Given: relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ with most relaxed task τ^*, new-goal-subset heuristic H_p, heuristic H_g
2: **function** ISSE($\mathcal{T}, H_p, H_g$)
3: $\mathcal{M} \leftarrow \{\}$ ▷ *map of MSGSs*
4: $\mathcal{F} \leftarrow \{\tau^* : \{(I^*, 0)\}\}$ ▷ *map of frontier states*
5: $T_p \leftarrow \{\}$ ▷ *processed tasks*
6: **while** $|T_p| < |\mathbb{T}|$ **do**
7: $\tau \leftarrow$ RANDOM$(\{\tau' \in \mathbb{T} \setminus T_p | pre(\tau') \subseteq T_p\})$ ▷ *next relaxed task*
8: $T_p \leftarrow T_p \cup \{\tau\}$
9: $\widetilde{M} \leftarrow \bigcup_{\tau' \in pre(\tau)} \mathcal{M}[\tau']$ ▷ *propagate MSGS*
10: $F \leftarrow \bigcup_{\tau' \in pre(\tau)} \mathcal{F}[\tau']$ ▷ *possible frontier*
11: $F \leftarrow \{(s, g_s) \in F \mid \neg\exists \tau'' \in pre(\tau) :$ APPLICABLE$(\tau'', \pi(s))\}$
12: $\mathcal{I} \leftarrow \{(s, g_s) \in F \mid$ APPLICABLE$(\tau, \pi(s))\}$ ▷ *actual frontier*
13: $\mathcal{M}[\tau], \mathcal{F}[\tau] \leftarrow$EXTEND$(\mathcal{I}, \tau^*, H_p(\tau^*), H_g(\tau^*), \tau, \widetilde{M},)$
14: $\mathcal{F}[\tau] \leftarrow \mathcal{F}[\tau] \cup F \setminus \mathcal{I}$ ▷ *add frontier states not reachable in τ*
15: **if** $G^{\text{soft}} \in \mathcal{M}[\tau]$ **then** ▷ *skip solvable tasks*
16: $T_p \leftarrow T_p \cup \{\tau' \in \mathbb{T} | \tau'$ is reachable from τ in $\mathcal{T}\}$
17: **return** $\{HIT(\overline{\mathcal{M}(\tau)}) \mid \tau \in \mathbb{T}\}$

For each task in $\mathcal{T}$, its MSGS $\mathcal{M}$ and the resulting search frontier $\mathcal{F}$ are stored. As for GSBnBp, the MSGS are propagated from less to more relaxed tasks (line 9). The frontier is initialized with the initial state of the most relaxed task τ^* (line 4). In each subsequent iteration, it is initialized with the frontier states of all predecessor tasks of τ that are reachable in τ (line 11-12). This reachability check is necessary because if $\tau \neq \tau^*$ there might be states that are frontier states for the predecessors of τ as well as for τ.

Then for each task τ we call EXTEND, to extend the search space from the frontier states $\mathcal{I}$ as far as possible and to collect all MSGS and frontier states of τ. Heuristic guidance and pruning follows the same approach as in GSBnB. However, the heuristic and the cost bound are not based on the current relaxed task τ (as in GSBnBp), but on the most relaxed task τ^*. Otherwise, states that might be reachable in more relaxed tasks could be pruned. The search frontier consists of the states that are generated in the search of τ but are not reachable in τ (line 23 - 26). To determine whether s' is reachable in τ, APPLICABLE tests if the action sequence $\pi(s')$ leading to state s' is applicable in the initial state of τ and whether cost$(\pi(s')) \leq b$.

Upon termination of ISSE, $\mathcal{M}$ contains the MSGS of all tasks. They can then be used to compute the MUGS of all task as described in Section 4.2.

PROPOSITION 19: CORRECTTNESS ISSE

Given a relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ ISSE computes the MUGS of all tasks in $\mathbb{T}$.

Algorithm 8 Iterative Search Space Extension (ISSE): EXTEND

1: Given: task τ, most relaxed task τ^*, new-goal-subset heuristic H_p, heuristic H_g, MSGS $\widetilde{M}$ of predecessor tasks and frontier states $\mathcal{I}$
2: **function** EXTEND($\mathcal{I}, \tau^*, H_p, H_g, \tau, \widetilde{M}$)
3: $\mathcal{G} \leftarrow \widetilde{M}$ ▷ *current maximal solvable goal subsets*
4: $g(s) \leftarrow \infty$ for all states $(s, x) \neq \mathcal{I}$ and $g(s) \leftarrow g_s$ for $(s, g_s) \in \mathcal{I}$
5: $\mathcal{O} \leftarrow \{(I, g(I)) \mid I \in \mathcal{I}\}$ ▷ *priority queue ordered by ascending* $H_g(s, b - g(s), \mathcal{G})$
6: $F \leftarrow \{\}$ ▷ *new frontier states*
7: **while** $|\mathcal{O}| > 0$ **do**
8: $s, g_s \leftarrow$ POP($\mathcal{O}$)
9: **if** $g(s) < g_s$ **then** ▷ *duplicate check*
10: **continue**
11: **if** $G^{\text{hard}} \cup G^{\text{soft}} \subseteq s$ **then**
12: **return** G^{soft}, F ▷ *all goals are solvable*
13: **if** $G^{\text{hard}} \subseteq s$ **then**
14: UPDATE($\mathcal{G}, s \cap G^{\text{soft}}$)
15: **for all** $a \in A(s)$ **do** ▷
16: $s' \leftarrow s[\![a]\!]$
17: $g_{s'} \leftarrow g(s) + c(a)$
18: **if** $g_{s'} \geq g(s')$ **then** ▷ *duplicate check*
19: **continue**
20: $g(s') \leftarrow g_{s'}$
21: **if** $g(s') > b \vee H_p(s, b - g(s'), \mathcal{G}) = \infty$ **then**
22: **continue** ▷ *prune* s' *if nothing new reachable*
23: **if** APPLICABLE($\tau, \pi(s')$) **then**
24: $\mathcal{O} \leftarrow \mathcal{O} \cup (s', g(s'))$
25: **else**
26: $F \leftarrow F \cup \{(s', g(s'))\}$ ▷ *update frontier*
27: **return** $\mathcal{G}, F$

Proof:

1. $H_p(\tau^*)$ is an admissible heuristic for any task τ' if $\tau' \sqsubseteq \tau^*$:
   τ^* is the most relaxed task of $\mathcal{T}$, thus for all tasks $\tau' \in \mathbb{T} : \tau' \leq \tau^*$. Since from $\Pi(\tau') \subseteq \Pi(\tau^*)$ follows that for all state $s \in \mathcal{S}^r(\Theta_{\tau^*})$ and for all goals $g \in G^{\text{soft}}$: $h^*_{\tau',g}(s) \leq h^*_{\tau^*,g}(s)$, where $h^*_{\tau,g}(s)$ is the perfect classical heuristic for task τ with no cost bound and $G = \{g\}$. Thus, pruning is safe following Proposition 7.

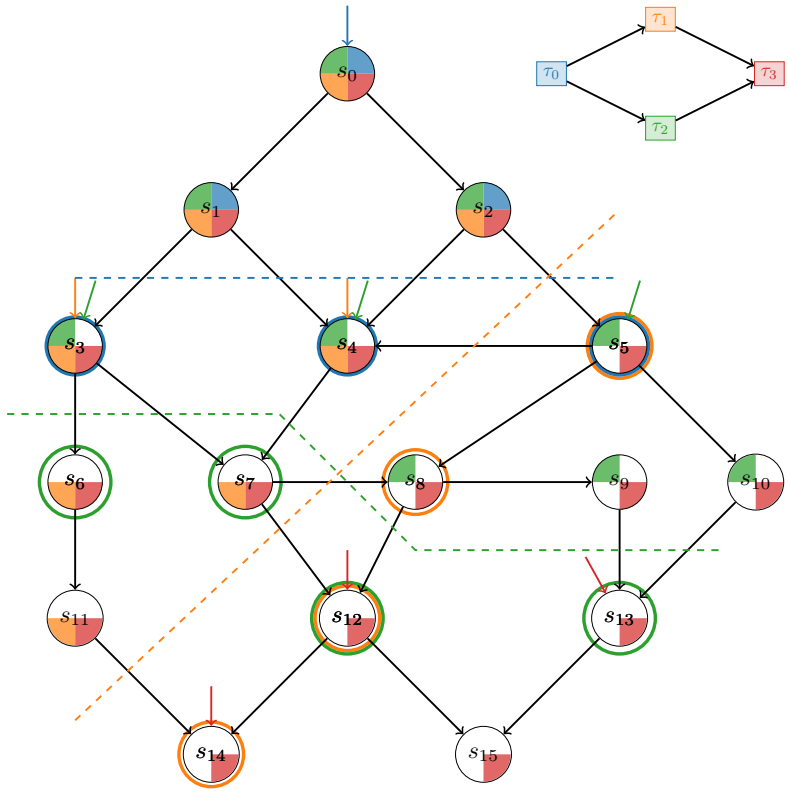
2. Proof via induction over $\mathcal{T} = (\mathbb{T}, E)$. The base case for task $\tau \in \mathbb{T}$ follows from Proposition 16. The induction step follows from Theorem 3.

□

Example 15 shows a schematic representation of such an iterative exploration of the state space.

EXAMPLE 15: SEARCH SPACE FOR ISSE

Let's consider an example with four tasks $\mathbb{T} = \{\tau_0, \tau_1, \tau_2, \tau_4\}$ such that $\tau_0 \sqsubseteq \tau_1$, $\tau_0 \sqsubseteq \tau_2$, $\tau_1 \sqsubseteq \tau_3$, $\tau_2 \sqsubseteq \tau_3$. Below the search space of the most relaxed task, τ_3 is depicted. Each state is colored with respect to its reachability. The frontier states for each task are labeled with a circle in the corresponding color. The "initial" states for each call to EXTEND are labeled with arrows.



For τ_0 only the states $\{s_0, s_1, s_2\}$ are reachable thus the states $\{s_3, s_4, s_5\}$ are frontier states. For τ_1 the search then starts from $\{s_3, s_4\}$ and explores states $\{s_6, s_7, s_{11}\}$, which results in the frontier $\{s_5, s_8, s_{12}, s_{14}\}$. Notice that state s_5 is in the frontier of τ_0 and τ_1 because it is also not reachable in τ_1. For τ_3 we only have to consider frontier states that are not yet covered by any less relaxed task, for example s_8 that is covered by τ_2.

5.2.3 THEORETICAL COMPARISON

The propagation of MSGS can improve pruning, which can benefit both GSBNBp and ISSE. Reusing the search space in ISSE reduces duplicate work, but states are only pruned based on the reachability in the most relaxed task, not the current task. We compare the overall number of generated states by each algorithm as a measure to determine whether

they are exponentially separated. As the baseline algorithm BASE, we consider GSBNBp without the propagation of MSGS.

DEFINITION 39: EXPONENTIAL SEPARATION

Let $\{T^n \mid n \in \mathbb{N}^+\}$ be a family of planning tasks of size (number of facts and actions) polynomially related to n and $S(X)$ the number of states generated by search method X. If there exists a family of planning tasks such that for search method X and Y, $|S(Y) - S(X)|$ is exponential in n, then X is **exponentially separated** from Y.

In the following we consider a family of resource constraint planning tasks (for the formal definition see Definition 45), in which a robot has to visit different locations on a map. The robot's movement is restricted by a resource ρ with domain $D(\rho) = \{0, 1, 2\}$. Additionally, some connections are initially locked, and the robot must first collect all n keys to unlock them. The robot can collect keys one at a time (without using any resources) if it is in the same location as the key. Since the keys can be picked up in any order, there can be exponentially many search states depending on n. The specific map configuration is defined for each example individually. We will consider tasks τ_1 and relaxed task τ_2 with initial resource value $\mathit{init}_\rho = 1$ and $\mathit{init}_\rho = 2$, respectively.

In the following examples, pruning is based on the h^2 heuristic [Haslum and Geffner, 2000] to estimate the reachability.

THEOREM 4: EXPONENTIAL SEPARATION FROM BASELINE

GSBNBp and ISSE are exponentially separated from BASE.

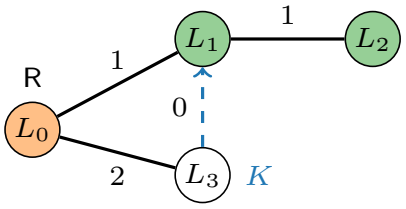


Figure 37: Map of separation of GSBNBp and ISSE from BASE, initial location: L_0, goal: visit L_1 and L_2; resource consumption per move is indicated by the edge labels. The dashed connection is initially locked, and the location annotated with K holds a set of n keys.

Proof:
Consider the map depicted in Figure 37. For τ_1 with $\mathit{init}_\rho = 1$ GSBNBp and the baseline generate 2 states (R at L_0 and L_1). ISSE, whose state space is based on τ_2 with $\mathit{init}_\rho = 2$, generates the same two states and 2 additional states (R at L_2 and L_3), which are not reachable and stored in the frontier. For both GSBNBp and ISSE MSGS $= \{\{L_1\}\}$ is propagated. For τ_2 with $\mathit{init}_\rho = 2$ in GSBNBp moving to L_3 is

pruned because no new locations are reachable from there. The same holds for ISSE. This leads to $2+3$ and $4+1$ states for GSBNBp and ISSE respectively. For the baseline, the reachability of L_1 is not propagated and moving to L_3 is not pruned. Thus, we get $2+3+2*2^n$ states, for picking up any combination of keys.

□

THEOREM 5: EXPONENTIAL SEPARATION GSBNBP AND ISSE

GSBNBp is exponentially separated from ISSE.

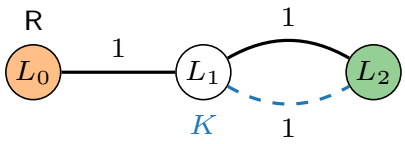


Figure 38: Map of separation of GSBNBp from ISSE, initial location L_0, goal: visit L_2; resource consumption per move is indicated by the edge labels. The dashed connection is initially locked, and the location annotated with K holds a set of n keys.

Proof:
Consider the map depicted in Figure 38. For τ_1 with $\mathit{init}_\rho = 1$, GSBNBp generates only one state. Moving to L_1 is pruned because h^2 recognizes that L_2 is not reachable with $\rho = 1$. In ISSE, the weaker constraint $\mathit{init}_\rho = 2$ prevents pruning of L_1 and picking up any combination of keys. Thus, $1 + 2^n$ reachable and 2^n (at L_2 with any combination of keys) unreachable states are generated. For τ_2 with $\mathit{init}_\rho = 2$ in GSBNBp visiting L_2 via the upper connection and extending the MSGS to $\{\{L_2\}\}$ leads to early termination. The same holds for ISSE. This results in $1 + 3$ states for GSBNBp and $1 + 2 * 2^n$ for ISSE.

□

5.3 COMPILATION OF ALLRELAXMUGS TO ALLMUGS

Until now, we have considered relaxation to be an external factor, one that modifies the planning task at hand. However, there is also another view, in which the planning task itself reflects only the hard constraints, such as the total possible resource capacity or the time windows given by natural circumstances. All actionable and useful soft constraints and possible relaxations of them are instead reflected by soft goals. In fact, we have already done so. In our running example, the soft goals that limit the usage of certain connections can be seen as a relaxation of the road map. This approach allows for the inclusion of relaxations in the conflict analysis without any special treatment. The following section describes how relaxations are encoded as soft goals and how the resulting MUGS have to be interpreted to yield AllRelaxMUGS.

5.3.1 Relaxation Soft Goals

In order to incorporate potential relaxations into the conflict analysis, it is necessary to encode them as soft goals. These soft goals serve to indicate which states are reachable within each relaxation.

Definition 40: Relaxation Soft Goals

Let τ be an OSP task and $\mathcal{T} = (\mathbb{T}, E)$ a relaxation graph for τ with most relaxed task τ^*.
A **relaxation soft goal** for task $\tau' \in \mathbb{T}$ is a soft goal $\gamma_{\tau'}$ that is satisfied in all states $\mathcal{S}^r(\Theta^{\tau'}_{\tau^*})$ and that is not satisfied by any state $\mathcal{S}^r(\Theta_{\tau^*}) \setminus \mathcal{S}^r(\Theta^{\tau'}_{\tau^*})$.
The set of all relaxation soft goals for $\mathbb{T}$ is denoted by Γ.

It is essential to emphasize *reachable* states. This implies the necessity of the expressive power of LTL_f formulas over actions and facts to define relaxation soft goals. Once the reachable states $\mathcal{S}^r(\Theta^{\tau'}_{\tau^*})$ for τ' are left, $\gamma_{\tau'}$ must be false for any subsequent state (see Example 16). These temporal soft goals are then compiled into a single goal fact as described in Section 4.3.2.

Example 16: Relaxation Soft Goals

Consider the following relaxations of the movement of two available rovers. In task τ no rover is used in relaxation τ_1 only rover R_1, in τ_2 only rover R_2 and in τ^* both. At first sight, the following relaxation soft goals, where l_1 and l_2 are the initial location of the rovers, make sense: $\Gamma = \{\gamma_\tau := R_1 = l_1 \wedge R_2 = l_2, \gamma_{\tau_1} := R_2 = l_2, \gamma_{\tau_2} := R_1 = l_1\}$. However, they are not only satisfied in all states reachable by action sequences not moving the respective rover, but also in all states where a rover returns to its initial location. Therefore, $R_1 = l_1$ needs to be globally enforced thus we need the following temporal soft goals: $\Gamma = \{\gamma_\tau := \Box(R_1 = l_1 \wedge R_2 = l_2), \gamma_{\tau_1} := \Box R_2 = l_2, \gamma_{\tau_2} := \Box R_1 = l_1\}$. For further examples see Section 5.4.1.

Given the set of relaxation soft goals Γ, the next step to compute AllRelaxMUGS, is to expand the soft goals of the most relaxed task τ^* by Γ. This preserves the original MUGS of τ^*.

Proposition 20: Extended Soft Goals and MUGS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ and $\tau' = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}'}, b)$ be OSP tasks that are equivalent except for the soft goals. If $G^{\text{soft}} \subseteq G^{\text{soft}'}$ then $\mathcal{G}^{\text{MUGS}}(\tau) \subseteq \mathcal{G}^{\text{MUGS}}(\tau')$.

Proof:
Follows from Definition 17 of MUGS.

□

Based on the MUGS of the with Γ extended task τ^*, the MUGS for each relaxation reflected by a soft goal in Γ can be determined. For each relaxation τ', the MUGS $\mathcal{M}_{\tau'}$ that are eventually resolved by a more relaxed task and the MUGS $\mathcal{M}_\emptyset$ that are still present in the most relaxed task τ^* must be considered. The former are characterized by a relaxation soft goal $\gamma_{\tau'}$, while the latter are not.

THEOREM 6: ALLMUGS TO ALLRELAXMUGS

Let τ be an OSP task, $\mathcal{T} = (\mathbb{T}, E)$ a relaxation graph for τ with most relaxed task τ^* and soft goals G_*^{soft} and Γ a set of relaxation goal facts for $\mathcal{T}$. Let τ_Γ be task τ^* with soft goals $G_*^{\text{soft}} \cup \Gamma$.
Then for all $\tau' \in \mathbb{T}$ the MUGS are given by:

$$\mathcal{G}^{\text{MUGS}}(\tau') = \min_{\subseteq}(\mathcal{M}_{\tau'} \cup \mathcal{M}_\emptyset)$$

with:

$$\mathcal{M}_{\tau'} = \{G \setminus \{\gamma_{\tau'}\} \mid G \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma), G \cap \Gamma = \{\gamma_{\tau'}\}\}$$
$$\mathcal{M}_\emptyset = \{G \mid G \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma) \wedge G \cap \Gamma = \emptyset\}$$

Proof:
Given an OSP task τ with hard goals G^{hard} and soft goals G^{soft} with state space Θ with goal states $\mathcal{S}_G^r$ and soft goal subset $G' \subseteq G^{\text{soft}}$, we denote the reachable goal states that satisfy G' by $RGS(\Theta, G')$ where $RGS(\Theta, G') \subseteq \mathcal{S}_G^r$ and for all $s \in RGS(\Theta, G')$, $G' \subseteq s$ holds.

$\mathcal{G}^{\text{MUGS}}(\tau') \subseteq \min_{\subseteq}(\mathcal{M}_{\tau'} \cup \mathcal{M}_\emptyset)$:

Let $C \in \mathcal{G}^{\text{MUGS}}(\tau')$.

a) $C \in \mathcal{G}^{\text{MUGS}}(\tau^*)$: With Proposition 20 we have $\mathcal{G}^{\text{MUGS}}(\tau^*) \subseteq \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$. Because C is minimal it follows $C \cap \Gamma = \emptyset$ and thus $C \in \mathcal{M}_\emptyset$. Since $C \in \mathcal{G}^{\text{MUGS}}(\tau')$ and $C \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$, there is no subset $C' \subsetneq C$ such that $C' \cup \{\gamma_{\tau'}\} \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$ and thus $\min_{\subseteq}$ will not remove C.

b) $C \notin \mathcal{G}^{\text{MUGS}}(\tau^*)$: From $C \in \mathcal{G}^{\text{MUGS}}(\tau')$ it follows $RGS(\Theta_{\tau^*}^{\tau'}, C) = \emptyset$, and for every $C' \subsetneq C$ there is a state $RGS(\Theta_{\tau^*}^{\tau'}, C') \neq \emptyset$. Since the goal state reachable in $\Theta_{\tau_\Gamma}^{\tau'}$ are the only goal states where $\gamma_{\tau'}$ is satisfied, $RGS(\Theta_{\tau_\Gamma}, C \cup \{\gamma_{\tau'}\}) = \emptyset$. Since $C \notin \mathcal{G}^{\text{MUGS}}(\tau^*)$,$RGS(\Theta_{\tau_\Gamma}, C) \neq \emptyset$. Thus, $C \cup \{\gamma_{\tau'}\}$ is unsolvable in Θ_Γ but every subset is. It follows $C \cup \{\gamma_{\tau'}\} \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$. Since $C \cup \{\gamma_{\tau'}\} \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$ there can

be no subset $C' \subsetneq C \cup \{\gamma_{\tau'}\}$ with $C' \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$ and thus C is not removed by $\min_\subseteq$.

$\mathcal{G}^{\text{MUGS}}(\tau') \supseteq \min_\subseteq(\mathcal{M}_{\tau'} \cup \mathcal{M}_\emptyset)$:

a) Let $C \in \mathcal{M}_{\tau'}$. This means $C \cup \{\gamma_{\tau'}\} \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$. Thus, $RGS(\Theta_{\tau_\Gamma}, C \cup \{\gamma_{\tau'}\}) = \emptyset$ and so $RGS(\Theta^{\tau'}_{\tau_\Gamma}, C)$ where. It follows that C is unsolvable in τ'. C is minimal, because $C \cup \{\gamma_{\tau'}\}$ is minimal and so for all $C' \subsetneq C : s \in RGS(\Theta_{\tau_\Gamma}, C' \cup \{\gamma_{\tau'}\}) \neq \emptyset$.

b) Let $C \in \mathcal{M}_\emptyset$. This means $C \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$. Thus $s \in RGS(\Theta_{\tau_\Gamma}, C) = \emptyset$ and hence also $RGS(\Theta^{\tau'}_{\tau_\Gamma}, C) = \emptyset$. It follows, C is unsolvable in τ'. If C is not minimal, then there is a $C' \in \mathcal{G}^{\text{MUGS}}(\tau')$ such that $C' \subsetneq C$. This means $RGS(\Theta^{\tau'}_{\tau_\Gamma}, C') = \emptyset$ and hence $RGS(\Theta_{\tau_\Gamma}, C') = \emptyset$. Thus, $C' \cup \{\gamma_{\tau'}\} \in \mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$ and $C' \in \mathcal{M}_{\tau'}$. Therefore, C would be removed by $\min_\subseteq$ because of C'.

□

$\mathcal{M}_{\tau'}$ are all MUGS for τ', which are resolved by a more relaxed task. $\mathcal{M}_\emptyset$ are all MUGS of τ_Γ that do not contain a relaxation soft goal. These MUGS are independent of the relaxed task and are unsolvable for all tasks in $\mathbb{T}$. However, they are *not* necessarily *minimal* for every less relaxed task. Therefore, an additional subset minimization of $\mathcal{M}_{\tau'} \cup \mathcal{M}_\emptyset$ is necessary. See Example 17 for an example.

EXAMPLE 17: ALLMUGS TO ALLRELAXMUGS

Let's consider our running example, this time with at most 11 units of energy. Assume we have given 10 relaxation soft goals $\Gamma = \{\gamma_i \mid 0 < i < 10\}$ where i refers to the available energy units for the rovers. This means γ_i is true in all states that can be reached with at most i units of energy. This results in 100 MUGS:

γ_i	0	1	2	3	4	5	6	7	8	9	10	$\emptyset$
#MUGS	6	6	6	5	5	10	12	10	10	13	6	11

With 2 or fewer units of energy, no data point can be uploaded, thus we have 6 singleton MUGS. With more than 2 units of energy soft goals become satisfiable. There are 11 MUGS not containing any relaxation soft goal. These are the conflicts that still exist in the most relaxed task with 11 units of energy.

To compute the MUGS for task τ_{10} we first take the 6 MUGS with γ_{10} and remove γ_{10}:

- {image-ice, image-crater3, x-ray-image-rock}
- {x-ray-image-rock, x-ray-image-crater3, soil-sample-ice}
- {image-ice, x-ray-image-rock, soil-sample-rock}
- {x-ray-image-crater3, soil-sample-rock, soil-sample-ice}
- {x-ray-image-rock, soil-sample-rock, soil-sample-ice}

- $\{\texttt{image-ice}, \texttt{x-ray-image-crater3}, \texttt{soil-sample-rock}\}$

Here 8 of the MUGS in $\mathcal{M}_\emptyset$ are supersets of a MUGS in $\mathcal{M}_{\tau_{10}}$, thus the subset minimization removes all of them. For example $\{\texttt{image-ice}, \texttt{image-crater3}, \texttt{x-ray-image-rock}\}$ is a stronger conflict and thus

- $\{\texttt{image-ice}, \texttt{image-crater3}, \texttt{x-ray-image-rock}, \texttt{soil-sample-rock}\}$
- $\{\texttt{image-ice}, \texttt{image-crater3}, \texttt{x-ray-image-rock}, \texttt{soil-sample-ice}\}$

are removed from $\mathcal{M}_\emptyset$ by the subset minimization.

The remaining 3 MUGS from $\mathcal{M}_\emptyset$ are:

- $\{\texttt{image-ice}, \texttt{x-ray-image-rock}, \texttt{x-ray-image-crater3}\}$
- $\{\texttt{image-ice}, \texttt{image-crater3}, \texttt{x-ray-image-crater3}, \texttt{soil-sample-rock}\}$
- $\{\texttt{image-ice}, \texttt{image-crater3}, \texttt{soil-sample-rock}, \texttt{soil-sample-ice}\}$

Together this results in $\mathcal{G}^{\text{MUGS}}(\tau_{10})$.

5.3.2 Non-Dominated MUGS

We have defined minimal unsolvable goal subsets as subset minimal, meaning that if we cannot remove any soft goal without resulting in a solvable goal subset, then we have a MUGS. However, if we consider relaxation soft goals, then subset-minimal is not the only relation that can be used. For two tasks τ_i and τ_j with relaxation soft goals γ_{τ_j} and γ_{τ_i}, if $\tau_i \sqsubseteq \tau_j$ and γ_{τ_j} is satisfied in a state s of τ_j then γ_{τ_i} is also satisfied in s (Definition 40). For example, in a resource-constraint task with relaxation soft goals $\gamma_{\leq 2}$ and $\gamma_{\leq 3}$ indicating that at most 2 and 3 units have been used, when $\gamma_{\leq 3}$ is satisfied, then $\gamma_{\leq 2}$ is also satisfied. Therefore, when $G \cup \{\gamma_{\leq 3}\}$ is unsolvable, then this also holds for $G \cup \{\gamma_{\leq 2}\}$. Consequently, when $G \cup \{\gamma_{\leq 3}\}$ is a MUGS, then all subsets $G \cup \{\gamma_{\leq x}\}$ with $x < 3$ could be considered non-minimal, since $\gamma_{\leq 3}$ could be replaced by the "weaker" $\gamma_{\leq x}$ without rendering the subset solvable.

Following the example above, we define dominance between soft goals as follows.

Definition 41: Soft Goal Dominance

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task. Let $g, g' \in G^{\text{soft}}$ be two soft goals. We say that g' **dominates** g, written $g \preceq g'$ if for all plans $\pi \in \Pi(\tau)$ if $g \in I[\![\pi]\!]$ it holds that $g' \in I[\![\pi]\!]$. We call $\preceq$ a **soft goal dominance relation**.

g' dominates g, if whenever g is satisfied then g' is satisfied as well. This reflects the relation between the relaxation soft goals in our example, if ≤ 2 units of fuel have been consumed so have ≤ 3 units.

PROPOSITION 21: RELAXATION SOFT GOALS AND DOMINANCE

Given a task τ and a relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ with most relaxed task τ^* and the relaxation soft goals Γ for $\mathbb{T}$. Then for all $\gamma_i, \gamma_j \in \Gamma, \tau_i, \tau_j \in \mathbb{T}$ if $\tau_i \sqsubseteq \tau_j$ then γ_j dominates γ_i ($\gamma_i \preceq \gamma_j$) in task τ^*.

Proof:
Based on Definition 40 γ_j is satisfied in all states $\mathcal{S}^r(\Theta^{\tau_j}_{\tau^*})$. From Proposition 17 follows that when $\tau_i \sqsubseteq \tau_j$ we have $\mathcal{S}^r(\Theta^{\tau_i}_{\tau^*}) \subseteq \mathcal{S}^r(\Theta^{\tau_j}_{\tau^*})$. Thus, γ_{τ_j} is satisfied in all states $\mathcal{S}^r(\Theta^{\tau_i}_{\tau^*})$, from which follows that for all $s \in \mathcal{S}^r(\Theta_{\tau^*})$ it holds $\gamma_i \in s \rightarrow \gamma_j \in s$. Thus, for all plans $\pi \in \Pi(\tau^*)$ if $g \in I[\![\pi]\!]$ then $g' \in I[\![\pi]\!]$.

□

Next, we combine the subset relation with the soft goal dominance relation.

DEFINITION 42: SUBSET DOMINANCE RELATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, $\preceq$ a soft goal dominance relation for G^{soft} and $G, G' \subseteq G^{\text{soft}}$. G' **subset-dominates** G, written $G \subseteq_D G'$, if $G \subseteq G'$, or if for all $g \in G$ there exists $g' \in G'$ such that $g \preceq g'$.

G' subset-dominates G if G is a subset of G' or if G contains only soft goals that are dominated by soft goals in G'. For example, we have $\{g_1, \gamma_{\leq 2}\} \subseteq_D \{g_0, g_1, \gamma_{\leq 2}\}$ but also $\{g_0, g_1, \gamma_{\leq 2}\} \subseteq_D \{g_0, g_1, \gamma_{\leq 1}\}$, where $\gamma_{\leq x}$ refers to the relaxation with at most x units of fuel can be consumed.

Based on this subset dominance relation, we can define *non-dominated* minimal unsolvable goal subsets that are minimal concerning subset minimality and the dominance relation between the soft goals.

DEFINITION 43: ND-MUGS

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and $\preceq$ a soft goal dominance relation for G^{soft}.
Subset $G \subseteq G^{\text{soft}}$ is a **non-dominated minimal unsolvable goal subset** (nd-MUGS) if $\tau' = (V, A, c, I, G^{\text{hard}} \cup G, \emptyset, b)$ is unsolvable and for all $G' \subsetneq_D G$ $\tau'' = (V, A, c, I, G^{\text{hard}} \cup G', \emptyset, b)$ is solvable.
By $\mathcal{G}^{\text{nd-MUGS}}(\tau, \preceq) := \{G \mid G \subseteq G^{\text{soft}}, G \text{ is a nd-MUGS}\}$ we denote the set of all nd-MUGS in τ for $\preceq$.

Given the dominance relation $\mathcal{G}^{\text{MUGS}}(\tau)$ can be reconstructed from $\mathcal{G}^{\text{nd-MUGS}}(\tau)$ and the other way around.

PROPOSITION 22: FROM ND-MUGSTO MUGS AND BACK

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and $\preceq$ a soft goal dominance relation for G^{soft}.

$$\mathcal{G}^{\text{nd-MUGS}}(\tau, \preceq) = \min_{\subseteq_D} \mathcal{G}^{\text{MUGS}}(\tau)$$
$$\mathcal{G}^{\text{MUGS}}(\tau) = \{G' \mid G \in \mathcal{G}^{\text{nd-MUGS}}(\tau, \preceq), \forall g \in G : \exists g' \in G' : g \preceq g'\}$$

Proof:
Follows from Definition 42.

□

To compute allRelaxMUGS for relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ the following steps are required:

(1) Define relaxation soft goals Γ for $\mathbb{T}$.

(2) Compute $\mathcal{G}^{\text{nd-MUGS}}(\tau_\Gamma, \preceq)$ where $\gamma_i, \gamma_j \in \Gamma : \gamma_{\tau_i} \preceq \gamma_{\tau_j}$ if $\tau_i \sqsubseteq \tau_j$

(3) Reconstruct $\mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$ from $\mathcal{G}^{\text{nd-MUGS}}(\tau_\Gamma, \preceq)$ (Proposition 22)

(4) Reconstruct allRelaxMUGS from $\mathcal{G}^{\text{MUGS}}(\tau_\Gamma)$ (Theorem 6)

To conduct a basic evaluation of the impact of the dominance relation on the allRelaxMUGS computation (see Section 5.5.3), we extended GLS, to compute $\mathcal{G}^{\text{nd-MUGS}}(\tau, \preceq)$ for a given task and dominance relation.

COMPUTATION OF ND-MUGS WITH GOAL LATTICE SEARCH

For nd-MUGS, the goal lattice is based on two relations: the subset relation between the soft goals and the soft goal dominance relation $\preceq$. Each node contains only soft goals that do not pairwise dominate each other.

DEFINITION 44: SUBSET-DOMINANCE GOAL LATTICE

Given a set of goals G and a dominance relation $\preceq$ between them the **subset-dominance goal lattice** is defined as $\mathcal{L}(G, \preceq) = (\mathcal{G}, \subseteq_D)$, where $\mathcal{G} = \{G' \subseteq G \mid g, g' \in G' : g' \not\succeq g \wedge g' \not\preceq g\}$.

To traverse the subset-dominance goal lattice we have to adapt the expansion functions for systematic weakening and strengthening as detailed in Algorithm 9.

For systematic weakening, we start with the subset of soft goals, that are not dominated by any other soft goal and either remove a soft goal or replace it with a dominated goal. For example, $\{g_0, g_1, \gamma_{\leq 2}\}$ can be weakened to $\{g_1, \gamma_{\leq 2}\}$ or $\{\gamma_{\leq 3}, g_0, g_1\}$. For systematic strengthening, we start with the empty set and either add a soft goal which does not

Algorithm 9 Expansion functions for systematic weakening and strengthening for nd-MUGS.

function INITSETSSYSW($G, \preceq$)
  return $\{G' \subseteq G \mid \forall g, g' \in G' : g' \not\succeq g \wedge g' \not\preceq g, \forall g' \in G' \neg\exists g \in G \setminus G' : g \succeq g'\}$
function EXPANDSYSW(G, solvable, $\mathcal{O}$, *Solvable*, *Unsolvable*, $\preceq$)
  if solvable **then** ▷ *Prune via solvability transitivity*
    return $\mathcal{O} \setminus \{G' \in \mathcal{O} | G' \subset_D G\}$
  else
    $\mathcal{G}_{\subset} \leftarrow \{G \setminus \{g\} \mid g \in G, \forall G' \in$ *Solvable* $: G \setminus \{g\} \not\subseteq_D G'\}$
    $\mathcal{G}_{\preceq} \leftarrow \{(G \setminus \{g\}) \cup g' \mid g \in G, g \preceq g', \forall G' \in$ *Solvable* $: (G \setminus \{g\}) \cup g' \not\subseteq_D G'\}$
    return $\mathcal{O} \cup \mathcal{G}_{\subset} \cup \mathcal{G}_{\preceq}$
function INITSETSSYSS($G, \preceq$)
  return $\{\emptyset\}$
function EXPANDSYSS(G, solvable, $\mathcal{O}$, *Solvable*, *Unsolvable*, $\preceq$)
  if $\neg$solvable **then** ▷ *Prune via unsolvability transitivity*
    return $\mathcal{O} \setminus \{G' \in \mathcal{O} \mid G' \supset_{\preceq} G\}$
  else
    $\mathcal{G}_{\subset} \leftarrow \{G \cup \{g\} \mid g \in G^{\text{soft}} \setminus G, \forall g' \in G : g' \not\succeq g \wedge g' \not\preceq g, \forall G' \in$ *Unsolvable* $: G \setminus \{g\} \subsetneq_D G'\}$
    $\mathcal{G}_{\preceq} \leftarrow \{(G \setminus \{g\}) \cup g' \mid g \in G, g' \preceq g, \forall G' \in$ *Solvable* $: G \setminus \{g\}) \cup g' \subsetneq_D G'\}$
    return $\mathcal{O} \cup \mathcal{G}_{\subset} \cup \mathcal{G}_{\preceq}$

dominate any other soft goal or replace a soft goal with a dominating soft goal. For example $\{g_0, \gamma_{\leq 2}\}$ can be strengthened to $\{g_0, g_1, \gamma_{\leq 2}\}$ or $\{g_0, \gamma_{\leq 1}\}$. In both expansion orders, sets that can be derived solvable/unsolvable because dominated/dominating sets have already been checked are excluded. All unsolvable nodes with only solvable children are the nd-MUGS.

5.4 PLANNING WITH RESOURCE AND TIME WINDOW CONSTRAINTS

In the following, we instantiate *relaxation explanations* for planning with resource and time window constraints. Resources of various forms are present in many planning domains and have a long tradition in planning [Koehler, 1998, Srivastava et al., 2001, Do and Kambhampati, 2014, Haslum and Geffner, 2001, Nakhost et al., 2012, Coles et al., 2013]. A resource is anything that has a limited amount and constrains the applicability of actions. We differentiate between *renewable* resources and *consumable* resources [Haslum and Geffner, 2001]. Renewable resources are resources that are not available during the execution of an action, but which regain their original value after the action is completed, e. g. the availability of a machine or a worker. Consumable resources, on the other hand, are consumed or produced when an action is applied. Examples include the fuel or energy resources of a truck or rover, as well as memory space or the space in a truck or elevator. Capturing an image consumes memory space, while transmitting the image frees up space again.

Actions are often completed over time, and certain conditions for their application may only be met during specific time frames, such as when the sun is shining or when workers

are available. Planning with different notions of time has been of interest for a long time [Smith and Weld, 1999, Haslum and Geffner, 2001, Fox and Long, 2003, Cushing et al., 2007, Do and Kambhampati, 2014, Jiménez et al., 2015].

Limited resources and time constraints not only make it difficult to find a plan, but are often the reason why there is no plan at all, and thus a common cause of conflicting soft goals. The relaxation of such resources or time constraints represents a natural relaxation for humans and is therefore well suited for explanations. For example, if your smartphone battery will not last for the trip, you will charge it before leaving the house, thus relaxing the resource by increasing the initially available amount. Similarly, if you cannot make it to an appointment, you call and ask whether you can come later, relaxing the time window by delaying the appointment. It is important to note that relaxations that are suitable for explanations should not affect the mechanics of the planning task. For example, allowing a rower to take photos even when it is dark would be a too drastic relaxation, leading to unusable plans. In addition, relaxations based on resources and time constraints enable gradual relaxations. For example, there are individual battery levels for energy, and there are also incremental increases for time windows. This allows for a trade-off between the amount of relaxation and the resolved conflicts.

In the following, we define OSP tasks with resource and time window constraints, replacing the overall cost bound.

PLANNING WITH CONSUMED RESOURCES

We consider consumable resources [Haslum and Geffner, 2001, Nakhost et al., 2012], that cannot be produced. This means the application of an action can only decrease, but not increase its value, i. e. refueling or recharging is not possible. Resources are limited to discrete domains, such that they can be compiled into classical planning. A planning task can have multiple resources related to different objects, such as the batteries of the different rovers.

DEFINITION 45: PLANNING TASK WITH CONSUMED RESOURCES

An OSP task with consumed resources is a task $\tau = (V, A, I, G^{\text{hard}}, G^{\text{soft}}, R)$, where R is a set of consumed resources.
A *consumed resource* ρ with domain $\mathcal{D}(\rho) = [0, \rho_{max}] \subset \mathbb{N}$ has an initial value $\mathit{init}_\rho \in \mathcal{D}(\rho)$ and a function $\delta_\rho : A \mapsto \mathbb{N}$ that maps each action to the amount of resource consumed by that action.
A complete assignment to $V \cup R$ is called a state.

The current resource values are represented by additional state variables. The applicability of the actions is based on the current resource value and the quantity consumed by the action. Applying action a in state s updates the variables in V according to their effect, and decreases the resource by $\delta_\rho(a)$ such that $s[[a]](\rho) = s(\rho) - \delta_\rho(a)$.

DEFINITION 46: STATE SPACE OSP TASK WITH CONSUMED RESOURCES

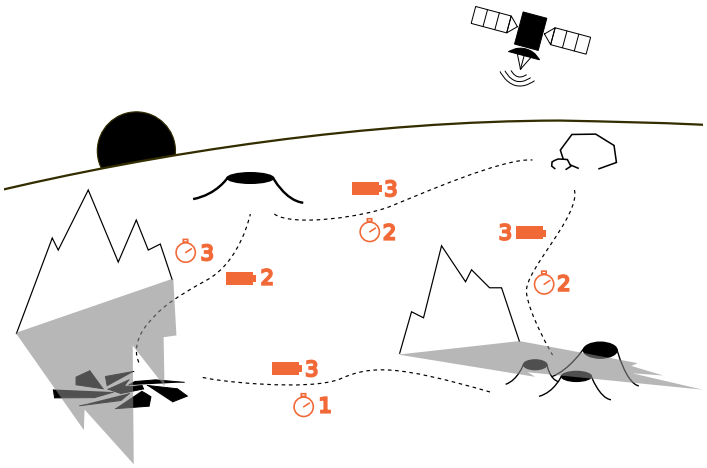
The **state space** of a OSP task with consumed resources $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, R)$ is the LTS $\Theta_\tau = (\mathcal{S}, L, T, I_\Theta, \mathcal{S}_G)$ where $\mathcal{S}, L, \mathcal{S}_G$ are defined as for an OSP task and

- $(s, a, s') \in T$, iff $s, s' \in S$, $a \in A$, $pre_a \subseteq s$, for all $\rho \in R : s(\rho) \geq \delta_\rho(a)$ and $s[\![a]\!] = s'$,
- the initial state $I_\Theta = I_\tau \cup \{v_\rho = \mathit{init}_\rho \mid \rho \in R\}$

The amount of resource ρ consumed by an action sequence π is given by $\mathit{con}(\pi) = \sum_{a \in \pi} \delta_\rho(a)$.

The extension of our running example with resources is given in Example 18.

EXAMPLE 18: PLANNING TASK WITH CONSUMED RESOURCES



In our running example, we include the battery ρ_b as a resource. Its initial value is $\text{init}_B = 8$, its maximal value is $B_{max} = 12$, and the energy consumed by each move action is given by the corresponding values in the map above. Additionally, collecting the data point also consumes energy:

- $\delta_{\rho_b}(\texttt{soil-sample}(l)) = 1$ for $l \in \{\texttt{rock}, \texttt{ice}\}$
- $\delta_{\rho_b}(\texttt{image}(l)) = 1$ for $l \in \{\texttt{crater3}, \texttt{ice}\}$
- $\delta_{\rho_b}(\texttt{x-ray-image}(l)) = 2$ for $l \in \{\texttt{crater3}, \texttt{rock}\}$

PLANNING WITH SIMPLE TIME WINDOWS

We consider planning with discrete time units, which includes an execution time for actions and constraints on *when* an action is applicable. We restrict ourselves to a concept of time

that can be compiled to classical planning. This includes discrete time units and no parallel execution of actions [Jiménez et al., 2015].

DEFINITION 47: PLANNING WITH SIMPLE TIME WINDOWS

An OSP task with simple time windows is a task $\tau = (V, A, I, G^{\text{hard}}, G^{\text{soft}}, \mathcal{W}, \delta_t, t_{max})$, where $t_{max} \in \mathbb{N}$ is the maximal number of time units, $\delta_t : A \mapsto \mathbb{N}$ a function that maps each action to its execution duration and $\mathcal{W}$ is a set of simple time windows.
A **simple (start) time window** is a tuple $W = (A_W, o, c)$ with $0 \leq o \leq c \leq t_{max}$. The application of the actions in $A_W \subset A$ is constrained by the opening time o and closing time c.
A complete assignment to $V \cup \{t\}$ with $\mathcal{D}(t) = [0, t_{max}]$ is called a state.

The passed time is represented by an additional state variable. The applicability of an action depends on the passed time and whether it is constrained by any of the time windows. When an action is applied, the time advances by the execution time of that action $\delta_t(a)$ such that $s[[a]](t) = s(t) + \delta_t(a)$.

DEFINITION 48: STATE SPACE OSP TASK WITH SIMPLE TIME WINDOWS

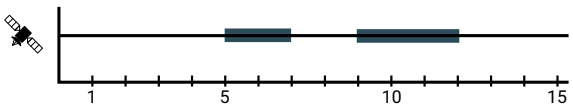
The **state space** of a OSP task with simple time windows $\tau = (V, A, I, G^{\text{hard}}, G^{\text{soft}}, \mathcal{W}, \delta_t, t_{max})$ is the LTS $\Theta_\tau = (\mathcal{S}, L, T, I_\Theta, \mathcal{S}_G)$ where $\mathcal{S}, L, \mathcal{S}_G$ are defined as for an OSP task and

- $(s, a, s') \in T$, iff $s, s' \in S$, $a \in A$, $pre_a \subseteq s$, for all $(A_W, o, c) \in \mathcal{W} : a \in A_W$ then $o \leq s(t) \leq c$ and $s[\![a]\!] = s'$,
- the initial state $I_\Theta = I_\tau \cup \{t = 0\}$

The execution duration of an action sequence π is given by $\mathit{dur}(\pi) = \sum_{a \in \pi} \delta_t(a)$ and the execution time point of action a in π by $\mathit{exec}(\pi, a) = \mathit{dur}(\mathit{prefix}(\pi, a))$.

The extension of our running examples with simple time windows is given in Example 19.

EXAMPLE 19: PLANNING TASK WITH SIMPLE TIME WINDOWS



In our running example, we introduce two time windows for uploading data

- $W_1 = (A_W, 5, 7)$
- $W_1 = (A_W, 9, 13)$

both with the same action set $A_W = \{\texttt{upload}(d) \mid d \in DP\}$.
The maximal time value is $t_{max} = 15$ and the time spent by each move action is given by the corresponding values in the map in Example 18. In addition, collecting the data points also spends time:

- $\delta_{\rho_b}(\texttt{soil-sample}(l)) = 2$ for $l \in \{\texttt{rock}, \texttt{ice}\}$
- $\delta_{\rho_b}(\texttt{image}(l)) = 1$ for $l \in \{\texttt{crater3}, \texttt{ice}\}$
- $\delta_{\rho_b}(\texttt{x-ray-image}(l)) = 1$ for $l \in \{\texttt{crater3}, \texttt{rock}\}$

5.4.1 Resource and Time Constraint Relaxations

There are various ways in which resource and time window constraints can be relaxed. In the following, we describe one option for each constraint type, the resulting relaxation graphs $\mathcal{T}$, and provide example explanations.

Resource Constraint Relaxations A task with consumed resources can be relaxed by increasing the initial resource value.

Definition 49: Resource-Relaxed Task

Let $\tau = (V, A, I, G^{\text{hard}}, G^{\text{soft}}, R)$ be a task with consumed resources. Then a resource relaxed task for resource $\rho \in R$ is defined as $\tau' = (V, A, I, G^{\text{hard}}, G^{\text{soft}}, R')$ where ρ is replaced by resource ρ' with $\mathcal{D}(\rho) = \mathcal{D}(\rho') = [0, \rho_{max}]$, $\delta_{\rho'} = \delta_\rho$ and $\mathit{init}_\rho \leq \mathit{init}_{\rho'} \leq \rho_{max}$.

Proposition 23: Resource-Relaxed Task

Let τ' be a resource-relaxed task of τ. Then τ' is a relaxed task of τ according to Definition 30 omitting the constraints on the action cost and cost bound, if soft goals are not defined over resources.

Proof:
$\Pi(\tau) \subseteq \Pi(\tau')$ because every action sequence $\pi = a_0 \cdots a_n$ applicable in I of τ is also applicable in I' of τ'. For all actions $a_i \in \pi$ with $\pi_i = \mathit{prefix}(\pi, a_i)$, $s = I[[\pi_i]]$, $s' = I'[[\pi_i]]$ and $c = con(\pi_i)$, a_i is applicable in s' because a_i is applicable in s, $s(V) = s'(V)$ and $\mathit{init}_\rho - c = s(\rho) < s'(\rho') = \mathit{init}_{\rho'} - c$. For all $\pi \in \Pi(\tau) : I[\![\pi]\!] \cap G^{\text{soft}} = I'[\![\pi]\!] \cap G^{\text{soft}}$ because in all intermediate states of π and thus also in the last state $s(V) = s'(V)$ and soft goals are not defined over resources.

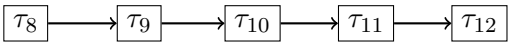
□

For task τ with resources R, the set $\mathbb{T}_\rho$ of all resource relaxed tasks for a resource $\rho \in R$, yields a relaxation graph $\mathcal{T}_\rho = (\mathbb{T}_\rho, E_\rho)$ for τ. Relaxation $\tau_i \in \mathbb{T}_\rho$ refers to the resource-

relaxed task with $\mathit{init}_{\rho_i} = i$. Since all $\tau_i \in \mathbb{T}_\rho$ are exclusively distinguished by init_{ρ_i}, we have $\Pi(\tau) \subseteq \Pi(\tau')$ iff $\mathit{init}_\rho < \mathit{init}_{\rho'}$ (Proposition 23), which results in a total ordering for $\mathbb{T}_\rho$. The edges of the relaxation graph $\mathcal{T}_\rho$ are given by $E_\rho = \{(\tau_i, \tau_j) \mid \tau_i, \tau_j \in \mathbb{T}_\rho, j = i + 1\}$. The task τ^* with $init_{\rho^*} = \rho_{max}$ represents the most relaxed task in $\mathbb{T}_\rho$. The resource relaxations for our running example are given in Example 20.

EXAMPLE 20: RESOURCE-RELAXED TASK

For our running example, we have four relaxed tasks for the battery $\mathbb{T}_{\rho_b} = \{\tau_i \mid i \in \{8, 9, 10, 11, 12\}\}$, where in τ_i the initial battery level is i. The relaxation graph is show below:



Reducing the resource requirements δ_ρ of an action a is another method of relaxing a task with consumed resources. Resources are exhaustible, and if action a appears once in the plan, this is equivalent to increasing the initially available resources. We use increasing resource availability as a proxy for any reduction in resource consumption.

Time Constraint Relaxations A task with simple time windows can be relaxed by increasing the time window, either by decreasing the opening time or by increasing the closing time.

DEFINITION 50: TIME-WINDOW RELAXED TASK

Let $\tau = (V, A, I, G^{\text{hard}}, G^{\text{soft}}, \mathcal{W}, \delta_t, t_{max})$ be a task with simple time windows. Then a time-window-relaxed task for time window $W = (A_W, o, c) \in \mathcal{W}$ is defined as $\tau' = (V, A, I, G^{\text{hard}}, G^{\text{soft}}, \mathcal{W}', \delta_t, t_{max})$ where W is replaced by time window $W' = (A_W, o', c')$ with $0 \leq o' \leq o \leq c \leq c' \leq t_{max}$.

PROPOSITION 24: TIME-WINDOW-RELAXED TASK

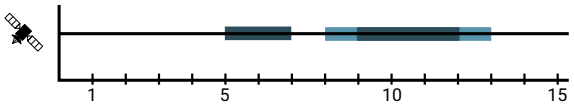
Let τ' be a time-window-relaxed task of τ. Then τ' is a relaxed task of τ according to Definition 30, omitting the constraints on the action cost and cost bound, if soft goals are not defined over time windows.

Proof:
$\Pi(\tau) \subseteq \Pi(\tau')$, because every action sequence $\pi = a_0 \cdots a_n$ applicable in I of τ is also applicable in I' of τ'. For all actions $a_i \in \pi$, $exec(\pi, a_i)$ is the same in both tasks and with $\pi_i = \mathit{prefix}(\pi, a_i)$, $s = I[[\pi_i]]$ and $s' = I'[[\pi_i]]$, a_i is applicable in s' because a_i is

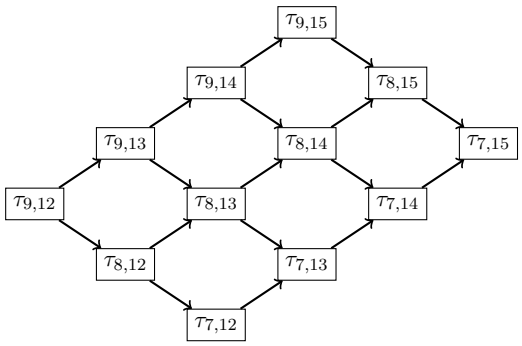
applicable in s, $s(V) = s'(V)$ and if $a_i \in A_W$ then $o' \leq o \leq \mathit{exec}(\pi, a_i) \leq c \leq c'$. For all $\pi \in \Pi(\tau) : I[\![\pi]\!] \cap G^{\text{soft}} = I'[\![\pi]\!] \cap G^{\text{soft}}$ because in all intermediate states of π and thus also in the last state $s(V) = s'(V)$ and soft goals are not defined over time windows. □

For task τ all time-window-relaxed tasks $\mathbb{T}_W$ for a time window $W \in \mathcal{W}$ compose a relaxation graph $\mathcal{T}_W = (\mathbb{T}_W, E_W)$. $\tau_{i,j} \in \mathbb{T}_W$ denotes the time-window-relaxed tasks with opening time i and closing time j. The subsumption relation of the intervals $[o', c']$ for time window W yields the partial ordering $\sqsubseteq$. The edges of the relaxation graph $\mathcal{T}_W$ are given by $E_W = \{(\tau_{i,j}, \tau_{n,m}) \mid \tau_{i,j}, \tau_{n,m} \in \mathbb{T}_W \wedge (m = j + 1 \vee n = i - 1)\}$. The task with $W' = (A_W, 0, t_{max})$ is the most relaxed task of $\mathcal{T}_W$. The time-window relaxations for our running example are given in Example 21.

EXAMPLE 21: TIME-WINDOW-RELAXED TASK



In our running example, we can relax either of the upload window. For the second window, we have 12 different relaxed tasks $\mathbb{T}_{W_U} = \{\tau_{i,j} \mid i \in \{7,8,9\} \wedge j \in \{12,13,14,15\}\}$, where in $\tau_{i,j}$ the opening time is at i and the closing time at j. The relaxation graph is show below:



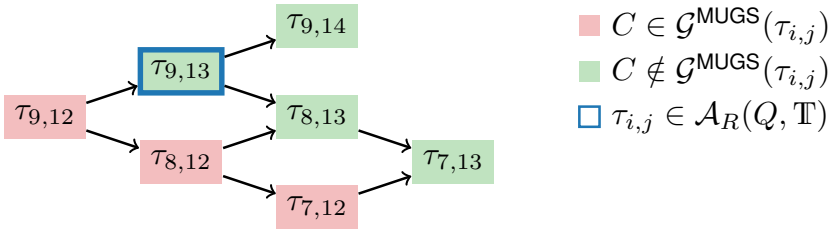
An alternative approach to relax a task with respect to time constraints is the reduction of the execution time of an action a by decreasing $\delta_t(a)$. However, in addition to affecting multiple time windows, handling the explosion of possible relaxed tasks is not trivial, which is why we leave this for future work.

An example for relaxation explanations based on time and resource constraint relaxations in given in Example 22.

EXAMPLE 22: RELAXATION EXPLANATION

First we consider the conflict $C = \{\texttt{image-ice}, \texttt{x-ray-image-rock}\}$ and the resource relaxed tasks $\mathbb{T}_{\rho_b} = \{\tau_i \mid i \in \{8, 9, 10, 11, 12\}\}$.
We have $C \in \mathcal{G}^{\text{MUGS}}(\tau_8)$ but $C \notin \mathcal{G}^{\text{MUGS}}(\tau_9)$, thus τ_9 is the minimally relaxed task for C. For the question "Why is C a conflict?" we thus get the answer: "Because to take the image of the ice surface and the x-ray image of the rock the rovers needs one additional unit of energy."

Next we consider the conflict $C' = \{\texttt{image-ice}, \texttt{soil-sample-rock}\}$ and the time window relaxations $\mathbb{T}_{W_U} = \{\tau_{i,j} \mid i \in \{7, 8, 9\} \wedge j \in \{12, 13, 14, 15\}\}$. Below the relevant part of the relaxation graph is depicted:



The minimal relaxed task is $\tau_{9,13}$ and thus the relaxation explanation $\mathcal{A}_R(C') = \{\tau_{9,13}\}$: "Because to upload the image of the ice surface and the soil sample of the rock, the second upload window has to close 1 time unit later."

RELAXATION SOFT GOALS FOR RESOURCE AND TIME CONSTRAINT RELAXATIONS

In the following, we define the relaxation soft goals for resource and time window relaxations we have introduced in Section 5.4.1.

In a relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ based on resource-relaxed tasks, the most relaxed task τ^* has the maximal initial resource value ρ_{max}. In a relaxed task τ_i with an initial resource value of i one can maximally consume i units. Therefore, the resource value v_ρ must not fall below $\rho_{max} - i$ to correspond to the resource available in τ_i.

PROPOSITION 25: RELAXATION SOFT GOALS FOR RESOURCE-RELAXED TASK

Let τ be a task with consumed resources R and $\mathcal{T} = (\mathbb{T}_\rho, E)$ a relaxation graph for τ based on resource-relaxed tasks for resource $\rho \in R$.
The relaxation soft goals are:

$$\Gamma_\rho = \{\gamma_{\tau_i} : \Box(v_\rho \geq \rho_{max} - i) \mid \tau_i \in \mathbb{T}\}$$

where i is the initial value of ρ in τ_i and v_ρ is the current value of ρ.

Proof:
Let τ be a OSP task with resource ρ and $\tau_i \in \mathbb{T}_\rho$ a resource relaxation of τ with respect to ρ with $\mathit{init}_\rho = i$. Let τ^* be the most relaxed task of $\mathbb{T}_\rho$ and $\gamma_{\tau_i} = \Box(v_\rho \geq \rho_{max} - i)$ be a temporal soft goal. For γ_{τ_i} to be a relaxation soft goal for τ_i, γ_{τ_i} must be satisfied in all $\mathcal{S}^r(\Theta^{\tau_i}_{\tau^*})$ (1) but not satisfied by any state in $\mathcal{S}^r(\Theta^{\tau_j}_{\tau^*}) \setminus \mathcal{S}^r(\Theta^{\tau_i}_{\tau^*})$, for all $i < j \leq \rho_{max}$ (2).

(1) All $\mathcal{S}^r(\Theta^{\tau_i}_{\tau^*})$ have a corresponding state in $\mathcal{S}^r(\Theta_{\tau_i})$ (Proposition 15). For all plans $\pi \in \Pi(\tau_i)$ we have $\mathit{con}(\pi) \leq i$. Since the δ_ρ stays the same, we have for all states $s \in \mathcal{S}^r(\Theta^{\tau_i}_{\tau^*})$, $s(v_p) \geq \rho_{max} - i$ and thus γ_{τ_i} is satisfied by all states in $\mathcal{S}^r(\Theta^{\tau_i}_{\tau^*})$.

(2) Let $s \in \mathcal{S}^r(\Theta^{\tau_j}_{\tau^*}) \setminus \mathcal{S}^r(\Theta^{\tau_i}_{\tau^*})$ where $s(v_p) \geq \rho_{max} - i$. This means there is a corresponding state in $s'' \in \mathcal{S}^r(\Theta_{\tau_j})$ such that $\mathit{con}(\pi) \leq i$ where π is the action sequence reaching s''. However, because we have $\tau_i \leq \tau_j$ from Proposition 23 follows $\Pi(\tau_i) \subseteq \Pi(\tau_j)$ and thus there is a corresponding state in $\mathcal{S}^r(\Theta_{\tau_i})$ and thus also in $\mathcal{S}^r(\Theta^{\tau_i}_{\tau^*})$. It follows $s(v_p) < \rho_{max} - i$.

□

For a time-window relaxed task, the specific window boundaries need to be satisfied. This can be done by checking that reaching a state by applying action a occurs at a time that satisfies the opening and closing times of the relaxed task.

DEFINITION 51: RELAXATION SOFT GOALS FOR TIME-WINDOW RELAXED TASK

Let τ be a task with simple time windows $\mathcal{W}$, and $\mathcal{T} = (\mathbb{T}_W, E)$ a relaxation graph for τ based on time-window-relaxed tasks for time window $W \in \mathcal{W}$.
The relaxation soft goals for time window $W = (A_W, o, c)$ are:

$$\Gamma_W = \{\gamma_{\tau_{i,j}} : \Box(\bigwedge_{a \in A_W} v_a \rightarrow i \leq t - \delta_t(a) \leq j) \mid \tau_{i,j} \in \mathbb{T}\}$$

where i, j are the opening and closing times of $\tau_{i,j}$, t is the current value of the time variable and variable v_a indicates the application of an action a according to the applied actions compilation in Definition 26.

Proof:
Let τ be a OSP task with resource ρ and $\tau_{i,j} \in \mathbb{T}_W$ a time-window relaxation of τ with respect to W with $o' = i$ and $c' = j$. Let τ^* be the most relaxed task of $\mathbb{T}_W$ and $\gamma_{\tau_{i,j}} = \Box(\bigwedge_{a \in A_W} v_{A_W} \rightarrow i \leq t - \delta_t(a) \leq j)$ be a temporal soft goal. For $\gamma_{\tau_{i,j}}$ to be a relaxation soft goal for $\tau_{i,j}$, $\gamma_{\tau_{i,j}}$ must be satisfied in all $\mathcal{S}^r(\Theta^{\tau_{i,j}}_{\tau^*})$ (1) but not satisfied by any state in $\mathcal{S}^r(\Theta^{\tau_{n,m}}_{\tau^*}) \setminus \mathcal{S}^r(\Theta^{\tau_{i,j}}_{\tau^*})$, for all $0 \leq n < i$ and $j < m \leq t_{max}$ (2).

(1) All $\mathcal{S}^r(\Theta^{\tau_{i,j}}_{\tau^*})$ have a corresponding state in $\mathcal{S}^r(\Theta_{\tau_{i,j}})$ (Proposition 15). For all plans $\pi \in \Pi(\tau_{i,j})$ we have for all $a \in A_W : i \leq \mathit{exec}(\pi, a) \leq j$. Since the δ_t stays the

same, this also holds for all $\Pi(\Theta^{\tau_{i,j}}_{\tau^*})$. Let $\pi \in \Pi(\Theta^{\tau_{i,j}}_{\tau^*})$ such that $\pi = \pi_0 a \pi_1$ for any $a \in A_W$. It follows that $i \leq \mathit{dur}(\pi_0) \leq j$ and thus $I_s[\![\pi]\!] = s$ with $i \leq s(t) \leq j$. This means after applying a when v_a is satisfied in $s' = s[\![a]\!]$ then, $s'(t) = s(t) + \delta(a)$ and thus $i \leq s'(t) - \delta_t(a) \leq j$ holds.

(2) Proof by contradiction: Let $s \in \mathcal{S}^r(\Theta^{\tau_{n,m}}_{\tau^*}) \setminus \mathcal{S}^r(\Theta^{\tau_{i,j}}_{\tau^*})$ such that $I_s[\![\pi a]\!] = s$ for any $a \in A_W$ and $i \leq s(t) - \delta_t(a) \leq j$ and thus $i \leq \mathit{exec}(\pi, a) \leq j$. This means there is a corresponding state $s' \in \mathcal{S}^r(\tau_{n,m})$ such that $I_{n,m}[\![\pi a]\!]$ and $i \leq \mathit{exec}(\pi, a) \leq j$. However, because we have $\tau_{i,j} \leq \tau_{n,m}$ from Proposition 23 follows $\Pi(\tau_{i,j}) \subseteq \Pi(\tau_{n,m})$ and thus there is a corresponding state in $\mathcal{S}^r(\Theta_{\tau_{i,j}})$ and thus also in $\mathcal{S}^r(\Theta^{\tau_{i,j}}_{\tau^*})$. It follows $\mathit{exec}(\pi, a) < i$ or $j < \mathit{exec}(\pi, a)$.

□

5.5 Computational Evaluation

Next, we evaluate both approaches to compute allRelaxMUGS for domains with resources or time window constraints. First, we focus on the task relaxation approaches, and then we return to the AllMUGS compilation approach using relaxation soft goals.

5.5.1 Experiment Setup & Benchmarks

Implementation

MSGS propagation search (GSBnBp) and iterative search space extension (ISSE) are implemented in Fast Downward[1] (FD) based on the exhaustive state space search (GSBnB) introduced in Section 4.2.2. The source code is publicly available[2]. Task relaxations are provided as additional input. The input definition is provided in the Appendix in Section B.1. For GSBnBp, the updates to the initial state are specified, and for ISSE the applicable actions in each relaxation are defined.

In addition, some adaptations were implemented in the Fast Downward translator. First, to circumvent the elimination of facts that are not delete-relaxed-reachable in the original task but could be reachable in a relaxed task, we added an option to consider multiple initial values for a variable during the reachability analysis. Second, the predicates defining the opening and closing time of time windows are static, meaning that they are not used in the effect of any action. Therefore, in the STRIPS-to-FDR transformation in the translator, these facts are not converted into variables. Instead, they are reflected by individual action instances, one for each possibility to satisfy the static preconditions. However, to allow for the opening and closing times to be modified for different relaxed tasks, they need to be represented as variables and preconditions of the grounded actions. To achieve this, we added an option to transform a subset of static predicates into FDR variables.

[1] https://github.com/aibasel/downward
[2] https://doi.org/10.5281/zenodo.14989835

ALGORITHMS

As we are using the same domains with encoded resource and time constraints, as for the temporal soft goals, h^{car} and h^{pot} cannot be used for pruning (see discussion Section 4.3.2). Thus, pruning is always based on h^{max}.

We evaluate the following algorithms:

baseline: As a baseline we use the straightforward approach of computing the MUGS for each relaxed task individually without propagating or reusing any information (basically GSBNBp without propagation of MSGS). As the search algorithm, we use GSBNB with and without pruning, referred to as BASE$^+$P and BASE.

GSBNBp: Iterative computation of MUGS with propagation of MSGS from less to more relaxed tasks with pruning Section 5.2.1.

ISSE: Iterative search space extension without pruning ISSE and with pruning ISSE$^+$P Section 5.2.2.

For the approach based on relaxation soft goals we use the following algorithms:

GSBNB Exhaustive state space search without pruning (GSBNB) and with pruning based on h^{max} (GSBNB$^+$P) Section 4.2.2.

GLS Goal lattice search with symbolic search for solvability test with expansion functions systematic weakening (SysW) and systematic strengthening (SysS) Section 4.2.1.

BENCHMARKS

The benchmark set is based on the same domains and instances used in Section 4.3.2 and is publicly available[3]. In each of the resource-constrained domains, Blocksworld, Nomystery, Rovers R and TPP, there are two resources R. Their initial value is set to the lowest value needed to be solvable for all simple goals (without additional temporal goals). For each task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, R)$ and each resource $\rho \in R$ we generated one benchmark instance, $\tau' = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, R')$ with $init_\rho = 0$ and $\rho_{max} = 2 * init_\rho$.

The time window constraint domains are Parent's Afternoon, Rovers T and Satellite. For each task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, \mathcal{W}, \delta_t, t_{max})$ and time window $W \in \mathcal{W}$ we generated one benchmark instance. For Parent's Afternoon and Satellite, each time window $W = (A_W, o, c)$ is relaxed between $[o, c]$ and $[0, t_{max}]$. In those domains, each time window constrains one activity, so it makes sense to increase the window to the largest possible size. For Rovers T, the time windows constrain when an upload to a relay satellite is possible. Since the model does not allow for multiple simultaneous uploads with overlapping time windows, we stop relaxing one time window once it matches the closing $c_<$ or opening $o_>$ times of earlier and later upload windows respectively. So, each time window $W = (A_W, o, c)$ is relaxed between $[o, c]$ and $[max\{c_<, 0\}, min\{o_>, t_{\max}\}]$.

[3] https://doi.org/10.5281/zenodo.14988342

For each $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, \cdots)$ resource or time window constrained, there are no hard goals $G^{\text{hard}} = \emptyset$ and the soft goals consist of a fixed set of simple goals (Blocksworld 8-13, Nomystery 5-8, Parent's Afternoon 4-8, Rovers R 5-7, Rovers T 4-8, Satellite 5-10, TPP 4-6) and from 1 up to 5 temporal soft goals. These, are the same 5 temporal soft goals used in the `all` benchmark set with LTL_f encoding (see Section 4.3.2).

The distribution of relaxed tasks across domains is depicted in Figure 39. Among the domains with resource constraints, TPP has the highest number of relaxed tasks, followed by Nomystery, Blocksworld, and finally Rovers R. In Parent's Afternoon and Satellite, the number of relaxations is comparable, ranging from about 10 to over 30, In Rovers T the majority of instances have fewer than 20 relaxations.

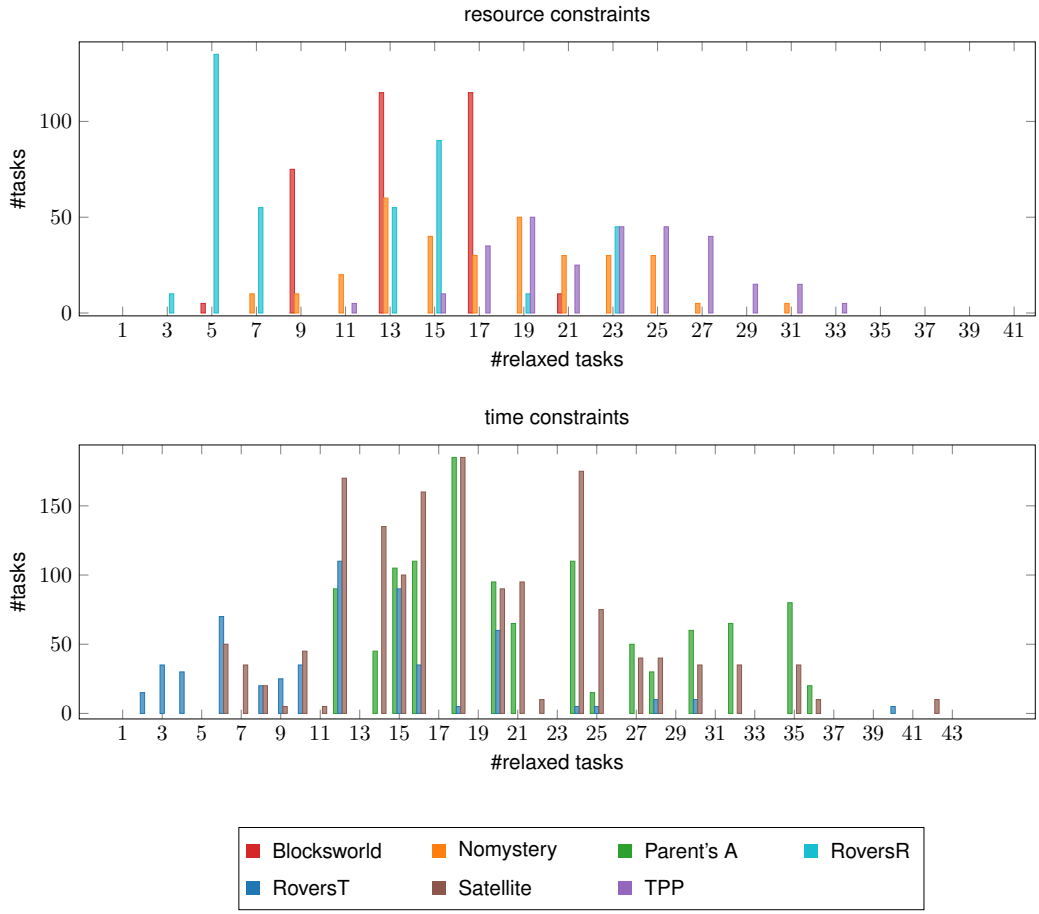


Figure 39: Distribution of relaxations in our benchmark set; x-axis number of relaxations, i. e.$|\mathbb{T}|$ for the relaxation graph $\mathcal{T} = (\mathbb{T}, E)$ based on either resource or time window relaxed tasks for each task τ, y-axis number of tasks.

For the instances for which the MUGS could be computed, the average number of MUGS over the relaxations is depicted in Figure 40. This analysis is based on instances for which complete data is available for the considered range of relaxations. For Blocksworld and TPP, the number of MUGS first increases when more resources are available, while for Nomystery and Rovers R, it decreases continuously. The zig-zag pattern in Blocksworld is due to the fact that most soft goals require two actions (pick up/unstack and put down/stack) to change

the truth value. Overall, the relaxations have a smaller impact on the time constraint tasks. The average number of MUGS in Parent's Afternoon changes slowly, suggesting that more options in scheduling activities has a limited impact on the conflicts. A notable resolution of conflicts is observed only after expanding the time windows for scheduling activities by approximately 7 to 8 units. In Rovers T and Satellite, a slight increase in MUGS is observed after relaxing the time windows by about 4 units.

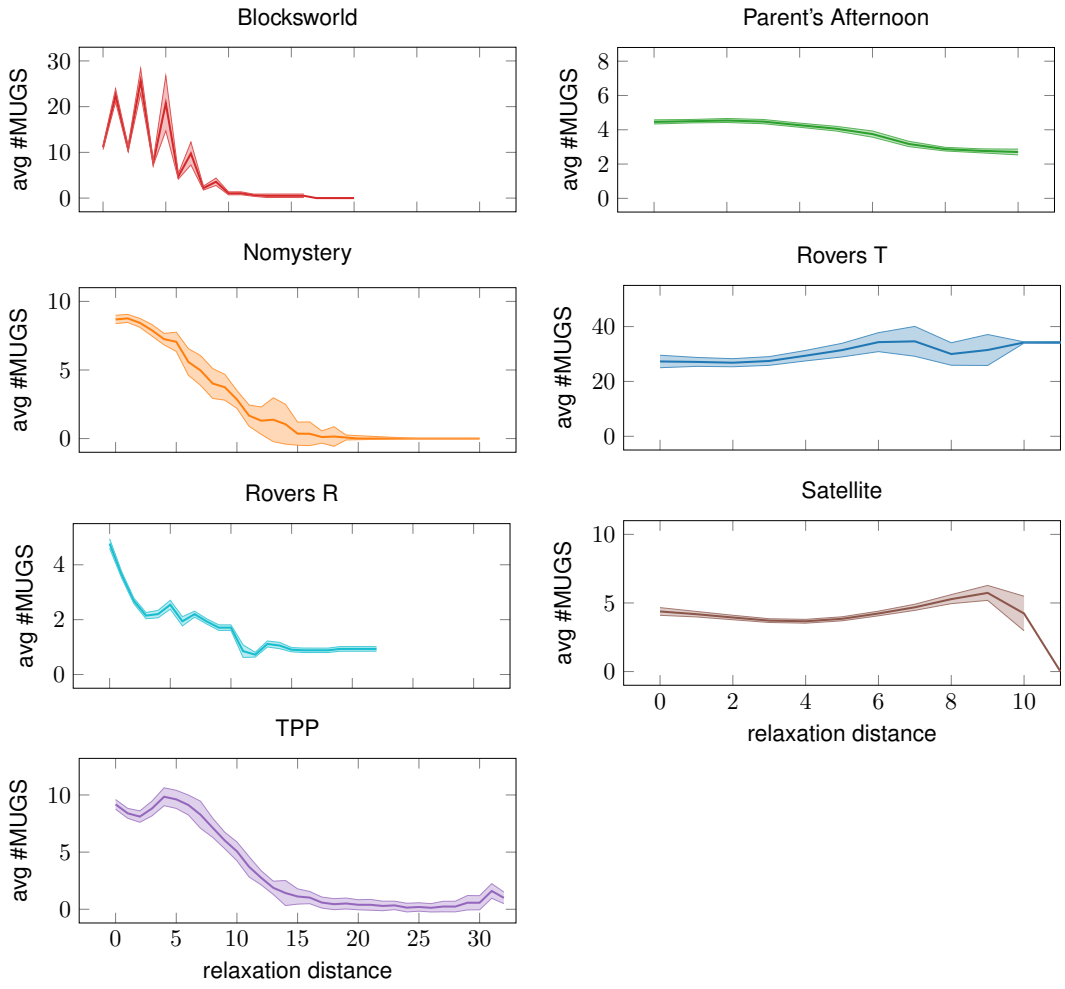


Figure 40: Number of MUGS over relaxations for instances solved by any approach; left: resource constraint domains, x value corresponds to $init_\rho$; right: time constraint domains, x value corresponds to the size difference of the relaxed time window to the original one. The shaded area represents the 90% confidence interval.

Next, we evaluate the approaches based on task relaxation and then compare the best configuration to the AllMUGS compilation approach using relaxation soft goals.

5.5.2 TASK RELAXATION

OVERALL COVERAGE

The coverage results are shown in Table 10. An instance is considered solved, when all MUGS for all relaxed tasks are computed. Pruning alone, without propagating the MSGS (BASE$^+$P vs BASE) , only improves the coverage in two domains (Nomystery and Rovers R). Comparing BASE$^+$P to GSBNBp shows that propagating MSGS increases the coverage significantly in all domains. This results in a better performance than BASE in 4 domains. Also, for ISSE$^+$P pruning significantly improves the performance, especially for Rovers R and the time-constrained domains. ISSE clearly has the advantage over GSBNBp in the time-constrained domains, while GSBNBp performance better in resource-constrained domains. Overall BASE performs best in 3 domains, ISSE$^+$P in 1 domain, GSBNBp in 3.

	domain	#	BASE	BASE$^+$P	GSBNBp	ISSE	ISSE$^+$P
resource	Blocksworld	320	**264**	179	184	170	185
	Nomystery	320	95	107	**126**	45	52
	RoversR	400	168	191	**215**	68	145
	TPP	290	184	156	**190**	45	56
time	Parent's A	1125	584	479	657	442	**788**
	RoversT	565	**545**	183	206	452	504
	Satellite	1560	**1099**	808	874	775	965
	sum	4580	**2939**	2103	2452	1997	2695

Table 10: Coverage (number of instances where allRelaxMUGS could be computed) comparison on resource and time window-constrained domains. The best result for each domain is highlighted in bold.

SCALING OVER RELAXED TASKS COMPLEXITY

The algorithms based on relaxed tasks compute allRelaxMUGS from the least to the most relaxed task, and can be seen as an anytime approach. Thus, even if not all relaxations could be addressed, we can still provide explanations based on the finished relaxations. Figure 41 depicts the relative number of instances where the MUGS could be computed over the number of relaxations. In domains with resource constraints, such as Blocksworld, Nomystery and TPP, as the number of relaxations increases, the coverage decreases and eventually reaches a plateau. This indicates that in some instances all soft goals are eventually solvable, allowing for the skipping of all more relaxed tasks. BASE consistently performs best in Blocksworld. For the remaining resource-constrained domains, the advantages of GSBNBp over ISSE$^+$P are clear across the entire range of relaxations. In domains with time constraints, the number of relaxations has a smaller impact on the coverage, decreasing more gradually with increasing relaxations. Eventually, a plateau is reached. This is not due to all soft goals being satisfied, but rather because further relaxations do

not result in more reachable states. There is a clear advantage of ISSE$^+$P over GSBnBp in Parent's Afternoon and in Rovers, which increases with the number of relaxations. In Satellite the coverage of ISSE$^+$P eventually drops below the coverage of GSBnBp.

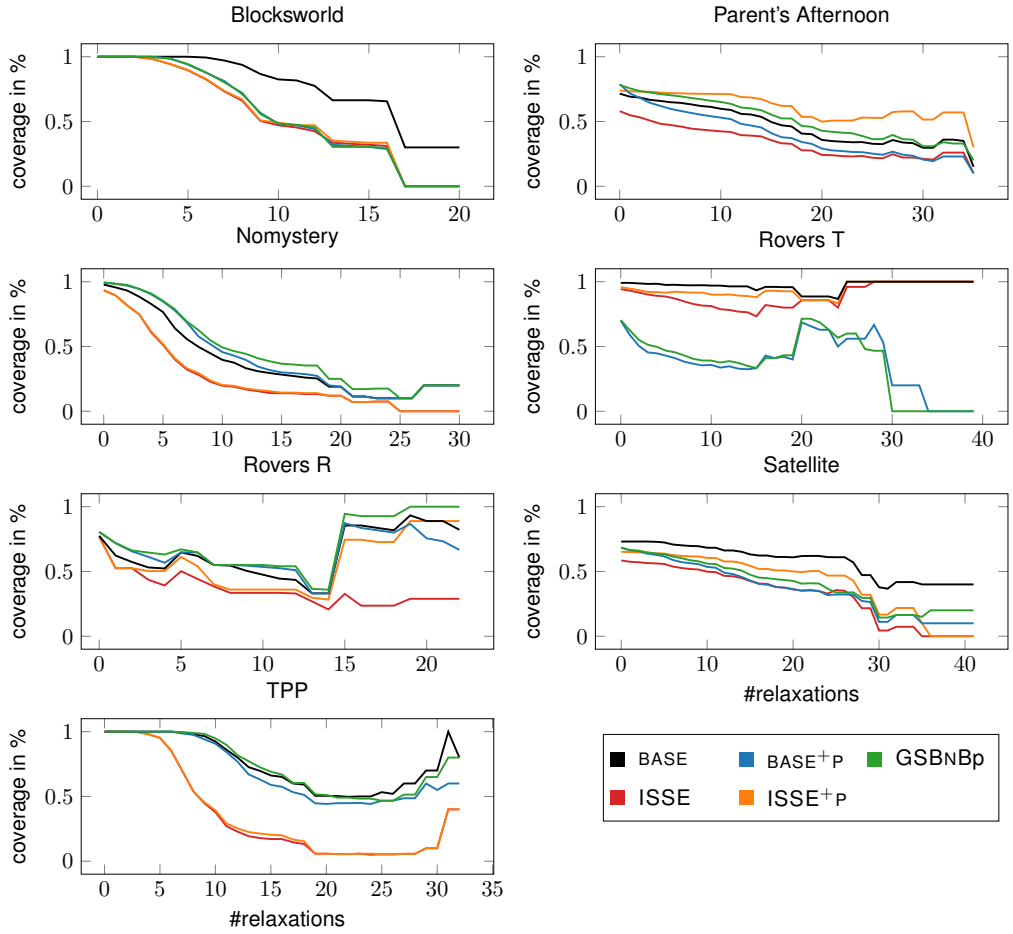


Figure 41: Relative number of finished relaxations, x-axis: number of relaxations, y-axis: relative number of relaxed tasks where MUGS could be computed.

The results indicate that 5 relaxations are generally feasible for domains with resource constraints. For Blocksworld and TPP, even 10 relaxations are still feasible in most cases. For the time-constrained domains, the results suggest that if the MUGS can be computed for the original task, then a larger number of relaxations can be handled.

Performance Analysis

The increase in reachable states caused by relaxing a time window is usually much smaller than for a resource. This is because increasing a time window only adds a few more time units at which an action $a \in A_W$ could start. However, as a is also constrained by all other time-dependent actions, there may not be many added reachable states. In contrast, relaxing a resource allows you to add new actions, such as traveling to more distant locations, and can also increase the number of actions that can be applied.

This is in favor of ISSE, as it exclusively considers the newly reachable states. A

comparison between the number of expansions each algorithm requires per relaxed task, as depicted in Figure 42, confirms this assumption. ISSE starts with more expansions due to the weaker pruning, yet the advantage of GSBNBp diminishes with increasing number of relaxations. This is particularly evident in time-constrained domains, where ISSE expands significantly fewer states than GSBNBp for a window size difference of 1 or more.

Figure 42 illustrates, that the pruning of states that cannot improve MSGS has a significant impact. The search space for BASE$^+$P is reduced by up to one order of magnitude in nearly all domains when compared to BASE. Additionally, the propagation of MSGS in GSBNBp further reduces the search spae. In domains with constrained resources this advantage increase with the number of relaxations, suggesting that GSBNBp is effective for larger number of relaxations. The comparison of ISSE and ISSE$^+$P indicates that pruning based on the most relaxed task has no effect in the resource-constrained domains. This is to be expected as for many instances eventually most soft goal subsets become solvable. In contrast, there is a substantial difference between ISSE and ISSE$^+$P in domains with time constraints. This indicates that, depending on the constraints and the most relaxed task, pruning against the most relaxed task can lead to a significant search space reduction. The following further comments on the results can be made. The number of expansions for BASE can drop because for some instances all soft goals become solvable, and thus the more relaxed tasks do not need to be addressed and are included with 0 expansions. The drop in GSBNB for Rovers R at 3 resource values is due to some cases where only one soft goal cannot be satisfied. The pruning function recognizes early on that this will remain unsolvable with the current fuel value.

Next, we compare the configurations in terms of their runtime, as shown in Figure 43. The comparison between BASE and BASE$^+$P (top left) indicates that pruning may not be a worthwhile investment if the MSGS are not propagated. However, if they are propagated (top right GSBNBp), then the time investment for Rovers R and parts of Parent's Afternoon and Nomystery pays off. A comparison of the runtime of ISSE and ISSE$^+$P reveals that ISSE either solves an instance within 100 sec or does not solve it at all. This is due to it exceeding the memory limit so quickly. Within this short time frame, pruning pays off for Satellite and partially in Parent's Afternoon. In the remaining time ISSE$^+$P solves about 600 additional instances, mainly in the time-constrained domains.

ISSE is more prone to memory issues than GSBNB. While, ISSE only expands the newly reachable states, it must still store the frontier states reachable by any not yet processed relaxation. In TPP, for instance, in the initial state of the most relaxed task, all possible combinations of $\mathtt{buy}$ actions are applicable, resulting in a large frontier already in the least relaxed task. Addressing this bottleneck is not trivial and thus part of future work.

As shown in Table 11, which analyzes the cause of an instance not being solved due to memory and time exhaustion, this issue is confirmed. As expected, the baselines with no pruning almost always exhaust the memory. For ISSE in the resource constrained domains and in Satellite, memory is the limiting factor. For GSBNBp, however, time is the deciding factor. This could be addressed by parallelizing tasks without a strict order.

While our methods are not yet outperforming the baseline in all domains, our analysis has identified a clear limiting factor. The more relaxations we consider, the more pronounced the impact of the information propagation underlying our approaches becomes.

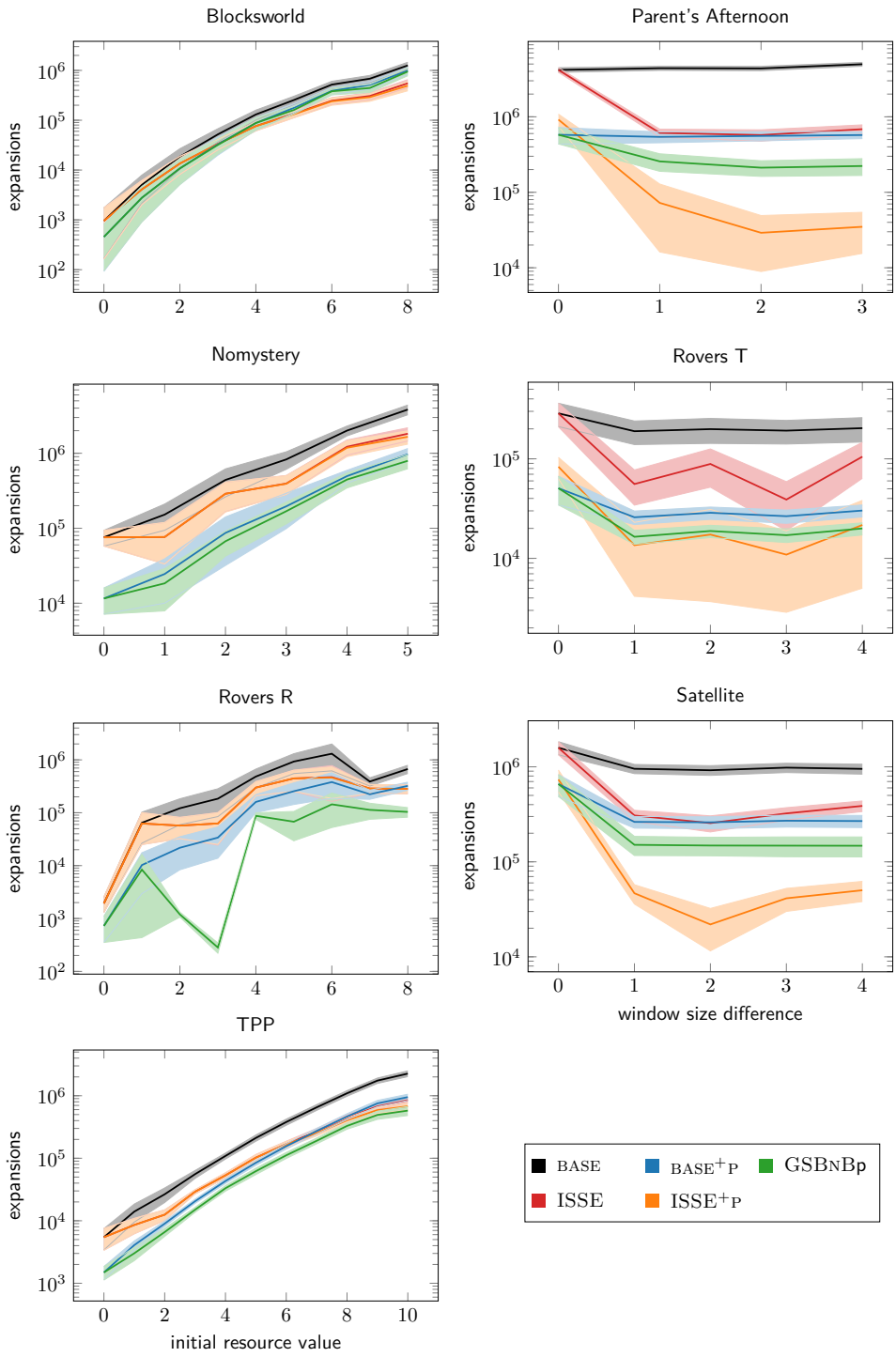


Figure 42: Comparison of the average number of expansions over commonly solved tasks. Shaded areas represent the 90% confidence intervals. Left: resource constraint domains, x value corresponds to $init_\rho$; right: time constraint domains, x value corresponds to the size difference of the relaxed time window to the original time window size. We only consider instances where all configurations have data for the full value range.

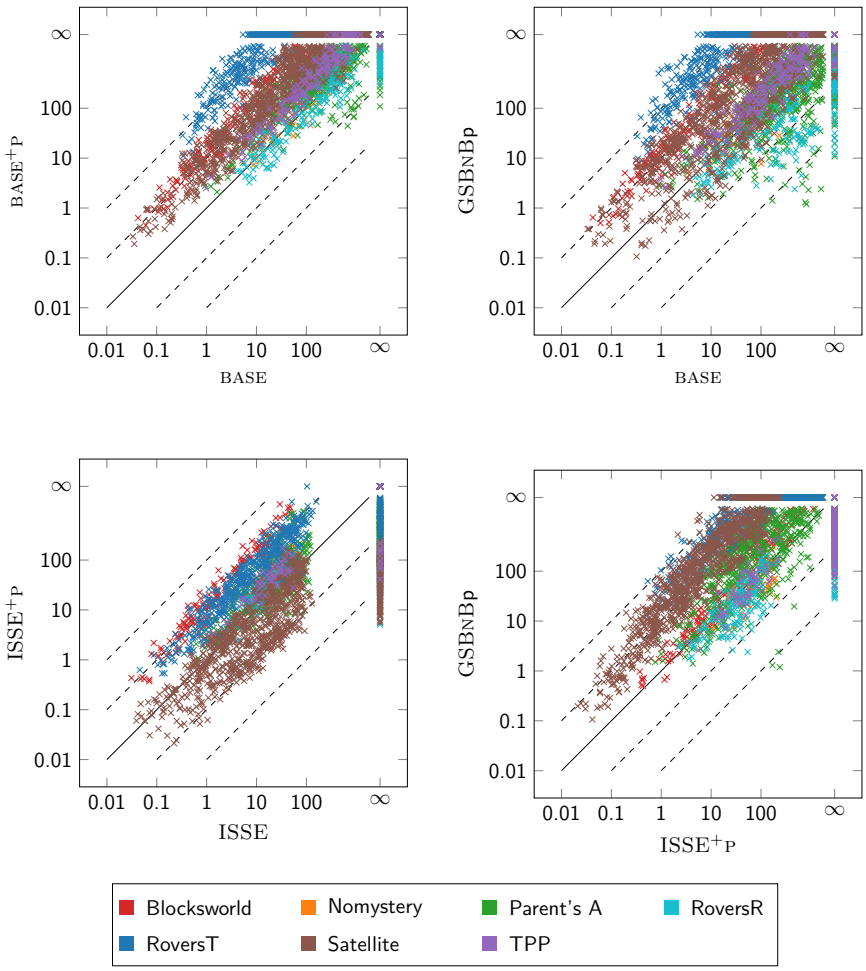


Figure 43: Instance-wise comparison of run time (in s). Not solved instances are assigned a runtime of ∞. Top: MSGS propagation and pruning, bottom left: with vs without pruning for ISSE and bottom right MSGS propagating vs iterative state space extension.

	domain	BASE	BASE$^+$P	GSBNBp	ISSE	ISSE$^+$P
resource	Blocksworld(320)	1	0	0	1	0.95
	Nomystery(320)	1	0.08	0.08	1	1
	RoversR(400)	1	0.7	0	1	1
	TPP(290)	0.98	0	0	1	1
time	Parent's A(1125)	0.75	0	0	1	0.45
	RoversT(565)	0.91	0	0	1	0.16
	Satellite(1560)	0.91	0	0	1	1

Table 11: Relative number of not solved instances in which the memory was exhausted, $1 - x$ is the relative number of not solved instances cases in which the time limit was reached.

5.5.3 RELAXATION SOFT GOALS

Next we analyze the performance of the AllMUGS compilation using relaxation soft goals. Instead of externally relaxing the task, each relaxation is represented by a soft goal that is only satisfied by states reachable within that relaxation.

The coverage results, in comparison to the task relaxation approach, are shown in Table 12. An instance is considered to be solved, when all MUGS for all relaxed tasks are computed or when all MUGS for the with relaxation soft goals augmented most relaxed task are computed. The results clearly show that, with a few exceptions in Blocksworld and Rovers R, the trivial relaxation soft-goal approach is not competitive. The large number of additional soft goals is not even feasible for GSBNB, which scales better than GLS in terms of the number of goals.

		relaxed tasks			relaxation soft goals			
	domain	BASE	GSBNBp	ISSE$^+$P	GSBNB	GSBNB$^+$P	SysS	SysW
resource	Blocksworld(320)	**264**	184	185	152	108	1	10
	Nomystery(320)	95	**126**	52	3	6	6	7
	RoversR(400)	168	**215**	145	0	65	100	102
	TPP(290)	184	**190**	56	5	7	0	0
time	Parent's A(1125)	584	657	**788**	0	0	0	0
	RoversT(565)	**545**	206	504	0	0	0	0
	Satellite(1560)	**1099**	874	965	0	0	0	0
	sum(4580)	**2939**	2452	2695	160	186	107	121

Table 12: Coverage comparison on resource and time window-constrained domains. Left: algorithms based on task relaxation, right: algorithms based on relaxation soft goals. The best result for each domain is highlighted in bold.

Theorem 6 does not consider the dependencies between the soft goals in Γ when computing AllRelaxMUGS. Section 5.3.2 discusses how these dependencies and the resulting non-dominated MUGS can be leveraged to accelerate the computation of AllRelaxMUGS.

We evaluate the computation of allRelaxMUGS via the computation of nd-MUGS, by considering the computation of all non-dominated MUGS for the tasks augmented with the relaxation soft goals τ_Γ ($\mathcal{G}^{\text{nd-MUGS}}(\tau_\Gamma, \preceq)$). This corresponds to step (2) of the complete procedure introduced in Section 5.3.2. The remaining steps are negligible in comparison. We used both expansion directions for GLS denoted by SysS$_\preceq$ and SysW$_\preceq$. Both use the dominance relation for the relaxation soft goals based on the corresponding relaxed tasks and $g \preceq g$ for every other soft goal. The input specifications are given in Appendix Section B.1, and the extension of GLSimplemented in symbolic Fast Downward[4] is publicly available.

The coverage results are presented in Table 13. An instance is considered solved, if either the MUGS for all relaxed tasks could be computed, or when all nd-MUGS/MUGS in the with the relaxation soft goals augmented most relaxed task could be computed.

[4] https://doi.org/10.5281/zenodo.14989835

			relaxed tasks			nd-MUGS		MUGS	
	domain	#	BASE	GSBNBp	ISSE+P	SysS$^{\Upsilon_1}$	SysW$^{\Upsilon_1}$	SysS	SysW
resource	Blocksworld	320	**264**	184	185	44	123	1	10
	Nomystery	320	95	**126**	52	88	95	6	7
	RoversR	400	168	215	145	**348**	327	100	102
	TPP	290	184	**190**	56	69	74	0	0
time	Parent's A	1125	584	657	788	**863**	753	0	0
	Rovers	565	**545**	206	504	503	475	0	0
	Satellite	1560	**1099**	874	965	757	948	0	0
	sum	4580	**2939**	2452	2695	2672	2795	107	121

Table 13: Coverage comparison on resource and time window-constrained domains. Left: algorithms based on task relaxation, right: algorithms based on relaxation soft goals via nd-MUGSand MUGS computation. The best result for each domain is highlighted in bold.

It is evident that the dominance relation between the relaxation soft goals is essential for the successful execution of the allRelaxMUGS calculation with relaxation soft goals. The number of solved instances increases significantly, even performing better than the approaches based on relaxed tasks, in 2 domains, Rovers R and Parent's Afternoon. Overall, time-constrained domains appear to be better suited to the approach with relaxation soft goals.

This indicates that the usage of relaxation goal facts is a competitive alternative to the task relaxation approaches. A more comprehensive analysis of the question which approach is preferable for which type of relaxation is left for future work.

5.6 Discussion

In the following we summarize the contributions and results presented in Chapter 5. We also discuss related work specific to this chapter and give an outlook on future work.

Summary

Once the conflicting soft goals C have been identified, a natural next questions is "Why are C in conflict?". We addressed this question by extending our explanation framework with *relaxation explanations*, which are explanations based on minimal relaxations of the planning task that resolve the conflict. This approach provides the user with information about why the conflict exists, as well as options for how it can be resolved.

To compute relaxation explanations, we must address the problem of allRelaxMUGS, i.e., computing all MUGS for a given set of relaxations. We presented two different approaches to address this problem, one based on two algorithms dealing specifically with task relaxations, and the other based on a compilation from allMUGS to allRelaxMUGS. In the former, we leverage information that can be propagated between relaxed tasks, such as MSGS

(GSBnBp) and the reachable search space (ISSE). We showed that GSBnBp and ISSE are exponentially separated from the baseline. While the trivial compilation approach proved to be infeasible, the inclusion of the dominance relation of soft goals reflecting the relaxations led to a promising alternative.

We instantiated relaxation explanations with relaxations based on resource and time window constraints, providing an example for relaxations well suited for explanations.

The empirical evaluation showed that relaxation explanations can be computed for a reasonable number of relaxations. There was no approach with the best overall performance, but a strong difference in performance depending on the domain and relaxation type, reflecting the variance of the problem structure. We demonstrated the performance improvement that can be achieved when using the right approach in conjunction with a suitable heuristic for each domain. Additionally, we highlighted the increasing effect of information propagation with increasing number of relaxations.

Related Work

In the following we discuss related work with respect to the use of relaxations for explanations and alternative approaches in planning to deduce why a task is unsolvable.

MUSes are used to explain why a set of constraints is unsolvable, and relaxations of constraints are used to suggest how to modify an unfeasible subset of constraints to make it feasible. In this context, relaxation often refers to removing constraints completely rather than relaxing the individual constraints. The first option is more in line with what we presented in Chapter 4, while the second approach is similar to what we discussed here. For instance, Senthooran et al. [2021] relax linear constraints (e. g. $x \leq b$) by using slack variables s, which reflect the extent of relaxation for each constraint (e. g. $x \leq b+s$). Different minimization criteria, allow one to identify a minimal relaxation with respect to the number of relaxed constraints or the overall slack. When multiple options exist for relaxing a task, e.g., due to limited resources or time, it becomes crucial to consider different objectives with respect to minimality. Elucidating the users preferences with respect to relaxation options and the affected conflicts is here considered part of the iterative process. Gupta et al. [2022b] provide explanations based on minimal relaxations to restore satisfiability for a new constraint added based on a user question. Using a predefined relaxation space they start with the loosest option for each constraint and tighten them iteratively until the constraint set is unsatisfiable. In our settings this approach resembles an online computation, addressing one specific user question rather than precomputing allRelaxMUGS. Starting from the user question “Why is C a conflict?”, C is added to the hard goals and then by iterating over the relaxations, from weakest to strongest or vice versa, the least relaxed tasks that are solvable are determined.

The scheduling system by Agrawal et al. [2020] provides information about constraint relaxations for no-scheduled activities. Their primary objective is to identify all unmet constraints of an activity and present them alongside the schedule to facilitate user review. Their analysis does not yet include any reasoning on the extent to which a constraint needs to be relaxed to schedule the activity. Yu et al. [2017] provide relaxation explanations for

over-constraint scheduling problems based on minimal execution duration shortening. In addition to explaining unsolvability, their approach is also used to analyze robustness. By finding the maximum feasible increase in execution, it is possible to determine how much leeway the tasks have in terms of execution delay.

By explaining why C is a conflict, we also address the question of why $G^{\text{hard}} \cup C$ is unsolvable. There has been some work in planning addressing the questions "Why is task τ unsolvable?", based on different motivations, as explanations for unsolvability of a specific instance, but also as debug support for model designers. Thus, useful explanations, i. e. changes to the task that can be implemented in a meaningful way, differ depending on the application. Sreedharan et al. [2019b] explain the unsolvability of a task by identifying necessary subgoals of relaxed solvable tasks, that are unachievable in the original task. This provides the user with easier to fulfill goals that cannot be solved, thus pointing to the actual cause of the unsolvability. However, due to the use of relaxations based on projections on subsets of variables, this approach is not suitable for quantifying the relaxation necessary to make the task solvable. An *excuse* for unsolvability, as defined by Göbelbecker et al. [2010], is a series of value changes in the initial state and additional objects to make a task solvable. These excuses may resemble relaxations, such as increasing the initial fuel value or adding a rover, but are not explicitly treated as such. Their approach does not address a specific conflict, but explains why the task in general is not solvable and focuses on pointing out errors in the model description. Another approach for modeling support by Lin et al. [2023] modifies the preconditions and effects of actions. Based on a valid plan and a flawed domain, they compute the minimum action changes to make the plan applicable. While this approach explains, why the current task/plan is unsolvable, the suggested task and model updates are not necessarily relaxations. Thus, they might introduce new conflicts.

Finally, the unsolvability certificates provided by the proof system of Eriksson et al., Eriksson et al. [2017, 2018] are not intended to be human-readable and do not provide information on how the task could be rendered solvable.

FUTURE WORK

User Study As was the case with conflict explanations, it is necessary to evaluate the usefulness of the relaxation explanations for human users. Do they adequately address the question "Why is C a conflict?" and do they help the user in finding better trade-offs? A user study following a similar set up as for conflict explanations would be appropriate.

Combination of Relaxations The combination of relaxations appears to be intuitive and useful. The user could define different relaxation graphs $\mathcal{T}_i$ each based on a specific way of relaxing the task, such as used waypoints, rovers, and resources and time windows. So far we are only considering individual relaxations. The response to "Why are Q in conflict?" considers each option to relax the task individually, leading to an answer like "To resolve conflict Q, you need to be able to traverse waypoints $\{l_0, l_1\}$ *or* use both rovers *or* increase the resource of rover R_1 by 3 units." It is possible that none of these relaxations is viable. However, a feasible approach might involve combining less relaxed tasks of each relaxation option. For instance, traversing waypoint l_0 *and* increasing the resource of rover R_1 by

1 unit could be an applicable relaxation. So the question arises how relaxations can be combined and how the resulting MUGS can be calculated efficiently.

Relaxation Selection Relaxing resource constraints by increasing the initial available amount and time constraints by increasing the time windows, is just one option. Alternatives include reducing the resource consumption or execution time of individual actions. One could also include relaxations of renewable resource like the number of rovers. However, handling the explosion of possible relaxed tasks is not trivial, especially if also the combination of different relaxation options should be considered. The applicability of relaxations and the preferences between them must be defined in collaboration with the user as part of the iterative planning process. It would therefore be advantages to automatically suggest possible relaxations and for the user to then select the relaxations to be taken into account.

Alternative Applications Instead of identifying a conflict and determining a resolution, the user could also adopt an alternative approach. The user could provide a relaxation and identify conflicts that could be resolved by applying it. For example, if there is an option to abort the previous mission early and finish with an overhead of resources, the question could be whether these additional resources would allow more data points to be collected in the next mission. Additionally, the robustness analysis as conducted by Yu et al. [2017], could be a valuable application. By tightening a task, e. g. reducing resources or time windows, it is possible to determine the extent of flexibility in terms of fuel reserves or execution delays.

Why? Questions never end, which means that the user may not understand why relaxation τ_r resolves conflict C and therefore asks “Why does τ_r resolve C?”. Consider for example the conflict $C = \{\texttt{take\_image\_crater}, \texttt{take\_sample\_ice}\}$, and the relaxation of the rover’s resource by two additional units. Taking the two samples involves multiple steps, such as initially driving to the corresponding locations. If conflict $C' = \{\texttt{visit\_crater}, \texttt{visit\_ice}\}$ exists, between visiting those locations, which is present in the original task and is also resolved by two additional energy units, then you know that not collecting the data is causing the conflict, but reaching the locations. An option to address the question “Why does τ_r resolve C?”, could be to use conflicts between "related" soft goals that are also resolved by τ_r. However, defining such soft goals is not trivial. One possible approach is to use landmarks [Hoffmann et al., 2004] of the soft goals in C in the relaxed task τ_r, similar to [Sreedharan et al., 2019b].

CHAPTER 6

LEARNING TEMPORAL GOALS

Our explanatory framework offers an analysis of conflicting soft goals. Ideally, these soft goals represent user preferences that extend to the aspects of the plan space in which they are interested. So far, we have assumed, that such a set of soft goals is provided by the user. However, it is often difficult for users to formalize their preferences. So the question is, how to simplify the process of preference elucidation.

There exist different approaches depending on the desired preferences and the possible interactions with the user. For an overview of planning with preferences we refer to Jorge et al. [2008]. For example, Lindsay et al. [2021], deal with the preference elucidation during planning by considering actions that reveal previously unknown user attributes. Mantik et al. [2022] introduce an elucidation framework for preferences represented by metrics over domain attributes. Based on diverse sets of plans from a set of small sample instances, they generate a set of questions consisting of pairwise comparisons of plans. The answers then provide a data set that is used to learn a linear preference function over the domain attributes. Using a similar setup, we explore the possibility of learning temporal soft goals from annotated sample plans. This is done in an iterative process, focusing on one temporal soft goal at a time. The individual steps are shown in Figure 44.

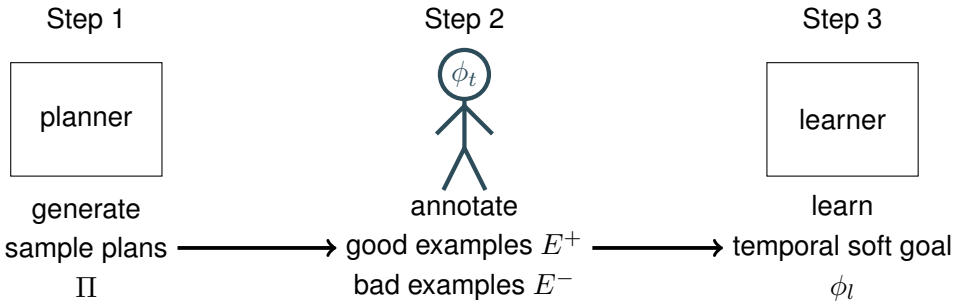


Figure 44: Process to learn one temporal soft goal ϕ_l based on annotated sample plans.

Step 1 is to generate a set of sample plans. Then, in **Step 2**, the user is asked to annotate them as good/bad plans with respect to the temporal soft goal ϕ_t they have in mind. The final **Step 3** is to learn a temporal soft goal ϕ_l that correctly identifies these positive/negative examples. These steps can be repeated to extract other temporal soft

goals until the user stops the process.

This approach itself is fairly straightforward, and its building blocks are known. For the planner in the first step, we use prior work to generate multiple plans for a given task focusing on different optimization and selection objectives [Katz et al., 2018, Speck et al., 2020, Katz and Sohrabi, 2020a]. For the learner in the last step, we leverage prior work for learning LTL_f formulas from sample traces [Neider and Gavran, 2018, Camacho and McIlraith, 2019, Kim et al., 2019]. Our contribution involves applying these techniques and empirically investigating their merits for soft goal learning.

To emulate the user in systematic experiments, we employ *hidden target formulas*. For each learned temporal soft goal, we fix a target formula ϕ_t. This formula is then used in Step 2, to annotate the sample plans. If a plan satisfies ϕ_t, it is a positive example, otherwise, it is a negative example. In practice, the hidden target formula ϕ_t will be inside the user's head. Emulating users in this form allows us to systematically evaluate different algorithmic methods.

We instantiate the hidden target formula ϕ_t with hand-made formulas based on PDDL3 preferences [Gerevini et al., 2009], as well as formula templates, which are often used in model checking [Manna and Pnueli, 1990, Dwyer et al., 1999, Menghi et al., 2019] and modeling business processes [van Der Aalst et al., 2009]. In Step 3, we use Camacho and McIlraith [2019] techniques, to learn a smallest LTL_f formula ϕ_l. The primary question we address is how to instantiate the planner in Step 1. Top-k methods with a focus on diversity are natural candidates, as they aim at producing k good-quality (in our context: short) yet qualitatively different plans. This makes intuitive sense for our purposes, since the example plans should cover different options for solving the task, but at the same time should not include obviously redundant behavior. We experiment with three different methods from the literature [Katz et al., 2018, Katz and Sohrabi, 2020a, Speck et al., 2020]. We furthermore, experiment with a simple randomized version of greedy best-first search using h^{FF} [Hoffmann and Nebel, 2001].

We evaluate the performance of different plan generation methods, by analyzing not only computational effort, but also the quality of the learned formula ϕ_l relative to the hidden formula ϕ_t, and the quality of plan examples in the sense of how many examples are needed to learn a high-quality formula.

Papers and Contributions This chapter is based on the paper:

Valentin Seimetz, Rebecca Eifler and Jörg Hoffmann
Learning Temporal Plan Preferences from Examples: An Empirical Study.
Proceedings of the 30th International Joint Conference on Artificial Intelligence (2021)

which in turn is partly based on the master's thesis by Seimetz [2020]. The author of this work acted as advisor of the master's thesis and proposed to evaluate the applicability of temporal preference learning for the introduced explanation framework. Seimetz provided the re-implementation of Camacho and McIlraith [2019] with the extension to learn all formulas of a given size. He also explored other methods to generate diverse plans and the learning of preferences without user input, not covered here.

The evaluation included here, which was conducted by the author of this work, encompasses an extension of covered templates, domains, plan generation approaches, and evaluated parameters.

6.1 Related Work & Building Blocks

The two main building blocks of our approach are the generation of plans and the learning of LTL_f formulas. In the following, we highlight related work in this area, while describing the approaches we decided to use in more detail.

6.1.1 Plan Generation

To provide a set of sample plans to the user, we must generate multiple plans for the given planning task. Since we assume that there is no prior knowledge of soft goals the user is interested in, the sample plan should provide a diverse selection. Ideally, these plans should cover different parts of the plan space, exhibiting different possibilities to reach the goal. Optimality with respect to plan cost is not a necessity; indeed, it is undesirable as it may exclude interesting plan options. However, this does not include redundant or repetitive behaviors, such as unnecessarily driving in circles. This increases the plan length without adding any valuable new options to the sample plan. In the following, we do not consider OSP tasks with a cost bound that could prevent specific behaviors from being reflected in sample plans. Instead, we use classical planning tasks without a cost bound and any additional encoded constraints such as resources or time. Additionally, we allow sub-optimal plans. Nevertheless, a bias toward small plan cost or length can make sense as cheap plans are generally preferable and short plans are easier to understand.

In the literature, two objectives are pursued when generating a set with k plans: *quality* [Katz et al., 2018, Speck et al., 2020] and *diversity* [Katz and Sohrabi, 2020b]. Quality generally refers to plan cost while for diversity between two plans π_0 and π_1 different metrics have been introduced. *Uniqueness similarity* [Roberts et al., 2014] identifies permutations and partial plans. *Stability similarity* [Fox et al., 2006, Katz and Sohrabi, 2020b] considers the ratio of common actions in π_0 and π_1 relative to all appearing actions. *State similarity* [Nguyen et al., 2012, Katz and Sohrabi, 2020b], on the other hand, focuses on the sequence of states resulting from the plan execution rather than on the similarity of actions. The similarity of two states is determined by the number of common variable values relative to the number of variables.

We explore four distinct plan generation techniques prioritize different objectives. All of these approaches can be run as an *anytime search*, enabling the generation of plans until a time limit is reached or a specified number of plans is found.

Top-K Planning First, we use *top-k planning* (`TopK`) [Katz et al., 2018, Speck et al., 2020]. Its default variant returns the best k plans in terms of cost. There are two primary approaches. First, an optimal plan is identified, and then, using an iterative method, a different plan with the same cost is computed or, if none exists, the maximal plan cost is increased. Katz et al. [2018] use an approach that iteratively calls an optimal planner and

modifies the planning task such that the found plan is not applicable in the next iteration. Speck et al. [2020] use symbolic planning. Once a goal state is expanded not just one, but all possible plans are reconstructed. The search space is then extended iteratively until k plans have been found or no more new states can be reached.

Top-K Planning with Permutation Filter In domains with independent objects, such as two trucks that can move independently, `TopK` frequently results in plans that are permutations of each other. These permutations do not reflect any temporal dependence between actions. Rather, they are an instantiation of the partial order defined by the preconditions and effects of the actions. As such, they do not represent another option for solving the planning task. To address this issue, we adopt a second approach that utilizes top-k planning with an additional filter. This filter removes plans that are permutations of already found plans, resulting in a set of plans that each have at least one distinct action (`TopKFil`).

Agile Diverse Planning Our third variant is *agile diverse planning* (`AgDiv`) by [Katz and Sohrabi, 2020a], which does not optimize plan quality but, similar to `TopKFil`, action diversity. It uses satisficing planning and iteratively computes new plans while forbidding all possible reorderings of already found plans.

Random As a simple approach generating different plans we also run a randomized version of h^{FF} [Hoffmann and Nebel, 2001] in greedy best-first search (`RNDhFF`). The randomization adds a positive random number to each heuristic value. As this approach does not guarantee to find different plans, the plans are filtered for uniqueness in a post-process.

6.1.2 Temporal Soft Goal Learning

There is substantial work in the area of learning LTL_f formulas from sample traces that we can build on [Neider and Gavran, 2018, Gaglione et al., 2021, Camacho and McIlraith, 2019, Kim et al., 2019] . The learner takes as input two sets of positive examples $E^+$and negative examples E^-. The output is a LTL_f formula ϕ that identifies the positive and negative examples. The objective is to maximize accuracy while minimizing formula size. Accuracy is measured by how accurately ϕ partitions the set of sample traces $E = E^+ \cup E^-$ into E^+ and E^-:

$$\alpha(\phi, E^+, E^-) = \frac{|\{\pi \in E^+ \mid \phi \vDash \pi\}| + |\{\pi \in E^- \mid \phi \nvDash \pi\}|}{|E^+| + |E^-|}$$

A formula ϕ is perfect if $\alpha(\phi, E^+, E^-) = 1$.

The approach by Kim et al. [2019] is based on probabilistic Bayesian models and relies on templates, meaning it cannot learn arbitrary LTL_f formulas. The probabilistic model enables robustness with respect to noise in the input data, which refers to traces that are not correctly sorted into E^+ or E^-. In our context, such noise would reflect plans incorrectly annotated by the user. This could be an interesting consideration for future work. For the

time being, we assume that the user either correctly annotates an example plan or does not annotate it at all, meaning that such noise does not exist.

Instead we follow other works [Neider and Gavran, 2018, Camacho and McIlraith, 2019], leveraging a SAT encoding to ensure perfect accuracy in identifying a smallest formula. This approach can use templates but does not rely on them, facilitating the learning of arbitrary formulas. In our evaluation, we use a modified re-implementation of the approach for LTL_f by Camacho and McIlraith [2019]. The learning process is an iterative process over the size of the learned formula. In each step, a SAT encoding of all LTL_f formulas of the given size and the input traces are generated. This SAT formula is satisfiable if one of the LTL_f formulas satisfies all positive examples and does not satisfy any negative example. The first satisfiable assignment the SAT-solver can find is then used to reconstruct the corresponding LTL_f formula. If the SAT encoding is unsatisfiable the size bound is increased.

6.1.3 TEMPORAL SOFT GOAL TEMPLATES

To instantiate the hidden target formulas we focus on templates based on commonly used temporal formulas in planning, model checking and business process modeling. In planning, LTL_f has been used to express user preferences [Gerevini et al., 2009] as in our case and also additional domain-dependent knowledge [Bacchus and Kabanza, 2000] to speed up the planning process. PDDL3 [Gerevini et al., 2009] extends the Planning Domain Definition Language (PDDL) with the feature to define temporal preferences. The language to define those preferences is based on LTL_f and supports a limited subset of nested modal operators defined as meta operators like for example `at-most-once`. It also extends to temporal planning, thus including operators referring to timed initial literals defined in PDDL2.2 [Hoffmann and Edelkamp, 2005]. Here we use the PDDL3 Preferences [Gerevini et al., 2009] that do not have numeric arguments. In the field of model checking [Manna and Pnueli, 1990, Dwyer et al., 1999, Menghi et al., 2019], LTL is used to define safety and liveness properties of systems. The most common language for modeling business processes is DECLARE [van Der Aalst et al., 2009]. We use their templates, translated to LTL_f by Bonassi et al. [2023].

Table 14 lists the formula templates we will consider for the construction of hidden target formulas in our empirical evaluation. These templates do not fully utilize the expressiveness of LTL_f, as user preferences often favor simpler temporal structures. This is evident, for example, by the fact that PDDL3 implements only a small part of LTL_f and that DECLARE, which is designed as an interface between LTL and human users, also limits the expressiveness.

6.2 ARCHITECTURE

We now discuss how to assemble these building blocks into an architecture for soft goal learning. We begin by briefly explaining the individual steps in our approach, highlighting the challenges and relevant special cases that can arise. Recall in what follows, we will assume a *hidden target formula*, denoted ϕ_t, inside the user's head.

	name/meaning	LTL_f formula	size
PDDL3	always(a)	$\Box a$	2
	sometimes(a)	$\Diamond a$	2
	sometimes-after(a,b)	$\Box(a \rightarrow \Diamond b)$	5
	at-most-once(a)	$\Box(a \rightarrow (a \mathsf{W} \Box\neg a))$	8
	sometimes-before(a,b)	$(\neg a \wedge \neg b) \mathsf{W} (a \wedge \neg b)$	10
Model Checking	safety	$\Box a$	2
	guarantee	$\Diamond a$	2
	persistence	$\Diamond\Box a$	3
	recurrence	$\Box\Diamond a$	3
	a before b	$\neg b \mathsf{U} a$	4
	mutual exclusion	$\Box\neg(a \wedge b)$	5
	precedence	$\Box$(a $\rightarrow\Diamond$b)	5
	simple obligation	$\Box a \vee \Diamond b$	5
	sequence	$\Diamond(a \wedge \Diamond b)$	5
	b forbids a	$\Diamond(b \rightarrow \Box\neg a)$	6
	persistent response	$\Diamond(a \wedge \bigcirc\Box b)$	6
	response	$\Box(a \wedge \bigcirc\Diamond b)$	6
	exception	$\Diamond a \rightarrow \Diamond(b \wedge \Diamond a)$	8
	stability	$\Diamond\Box a \wedge \Box(a \rightarrow \Box a)$	9
DECLARE	existence(a)	$\Diamond a$	2
	absence(a)	$\neg\Diamond a$	3
	choice(a,b)	$\Diamond a \vee \Diamond b$	5
	co-existence(a, b)	$\Diamond a \leftrightarrow \Diamond b$	5
	responded-existence(a, b)	$\Diamond a \rightarrow \Diamond b$	5
	response(a, b)	$\Box(a \rightarrow \Diamond b)$	5
	absence2(a)	$\neg(\Diamond a \wedge \bigcirc\Diamond a)$	7
	precedence(a, b)	$(\neg b \mathsf{U} a) \vee \Box\neg b$	8
	exclusive-choice(a,b)	$(\Diamond a \vee \Diamond b) \wedge \neg(\Diamond a \wedge \Diamond b)$	12

Table 14: LTL_f templates used as hidden target formula to simulate the user. We define the size of a formula as the number of sub-formulas.

STEP 1 PLAN GENERATION

In the first step, we generate a set E of plans for the given planning task, using one of the introduced approaches: `TopK`, `TopKFil AgDiv`, `RNDhFF`. Given that the target formula ϕ_t is hidden, the plan generation cannot be tailored to generate positive and negative examples for ϕ_t. Therefore, we simply consider the first n plans generated. We will experimentally explore the impact of the parameter n.

An important complication is that the learning step requires at least one positive and one negative example, which may not be given in the first n plans. In this case one must either give up or increase the value of n. However, the hidden user formula ϕ_t may be a tautology or unsatisfiable in the planning task, i. e., may be true (or false) in all plans. In practice, we

are unable to verify this. Note though that tautological or unsatisfiable soft goals are not meaningful in a conflict analysis. The former is not part of any MUGS, while the latter is a singleton MUGS. Presumably, users will possess sufficient knowledge about the domain and task at hand to formulate meaningful soft goals. In our experiments, we consider only soft goals, for which positive and negative examples exist.

STEP 2: PLAN ANNOTATION

We provide the set of example plans E to the user and request that they annotate the plans with respect to their hidden target formula ϕ_t as positive E^+ and negative E^- examples. The number of plans n the user has to annotate, should be as small as possible. We will evaluate empirically how many plans are necessary to learn ϕ_t. In practice, one could interleave plan annotation and formula learning until the user is satisfied with the result. In our setting here, this corresponds to analyzing learning performance as a function of n.

It is not necessary to annotate all plans in E. The user can select any subset of positive and negative examples that most clearly resemble the desired behavior ϕ_t. We assume that the user does not make any mistakes in the annotation, i. e. that ϕ_t has a perfect accuracy on E^+ and E^-.

STEP 3: LEARNING

Given E^+ and E^-, we call the learner to obtain the set Φ_l of smallest LTL_f formulas with a perfect accuracy. We use formula size as a proxy for quality, thus only considering the smallest perfect formulas. This is commonly done [Neider and Gavran, 2018, Gaglione et al., 2021, Camacho and McIlraith, 2019, Kim et al., 2019] when no preference function for different formulas is given. However, one has to keep in mind, that there are temporal soft goals, that have straightforward natural language descriptions, but the respective LTL_f formula is quite large, for example *at-most-once* ($\Box(a \rightarrow (a \,\mathsf{W}\, \Box\neg a))$) with a size of 8. In order to not disadvantage such penalize goals, commonly used templates could be introduced as meta-operators, which would not lead to an increase in formula size. This is an interesting approach for future work.

As ϕ_t is not known, learned formulas Φ_l can not be filtered further without additional user input. Therefore, all formulas in Φ_l are forwarded to the user for inspection.

There are five distinct ways in which the learned formulas Φ_l and the target formula ϕ_t can be related, each of which can be used differently within our explanation framework.

DEFINITION 52: LEARNED FORMULA AND TARGET FORMULA RELATION

Given a planning task τ with plans Π, the formulas $\phi_l \in \Phi_l$ are related to the target formula ϕ_t in exactly one of the following ways:

(a) we learn the same formula: $\phi_l = \phi_t$

(b) ϕ_l is equivalent to ϕ_t: $\Pi \models \phi_t \rightarrow \phi_l \wedge \Pi \models \phi_l \rightarrow \phi_t$

(c) ϕ_l is an over-approximation of ϕ_t: $\Pi \models \phi_l \rightarrow \phi_t$

(d) ϕ_l is an under-approximation of ϕ_t: $\Pi \models \phi_t \rightarrow \phi_l$

(e) no direct relation, i. e., none of (a)–(d) holds.

Note that for cases (b) to (d) we consider Π-entailment (Definition 12). $\Pi \models \phi_t \rightarrow \phi_l$ means that ϕ_t entails ϕ_l based on Π, i. e. $\{\pi \mid \pi \in \Pi, \pi \models \phi_t\} \subseteq \{\pi \mid \pi \in \Pi, \pi \models \phi_l\}$. This does not necessarily mean that ϕ_t implies ϕ_l based on the LTL_f semantics.

In the ideal case (a), the user receives the expected result. In the worst case (e), the user is presented with a set of unrelated formulas.

The intermediate cases are more difficult to judge. At first glance, equivalence (b) appears unproblematic. However, depending on how similar, syntactically and semantically, ϕ_l and ϕ_t are, the user may face difficulties in recognizing the equivalence. One has to keep in mind that the formulas are equivalent based on Π-entailment. In our experiments, we observed surprising equivalent formulas, that identified subtle dependencies in the planning task (see Example 23). On a positive note, this form of dependency identification presents an alternative application of our techniques. The plan annotation and formula learning are then used to automatically identify new formulas that relate in particular ways to previously identified preferences. We illustrate this potential alternative application of our techniques in the future work Section 6.4.

The usefulness of over/under-approximations (c) and (d) of ϕ_t also highly depends on their similarity to ϕ_t. A useful result would for example be $\phi_l = \Box a$ given the target formula $\phi_t = \Box(a \vee b)$, or in general if $\phi_l \rightarrow \phi_t$ based on the LTL_f semantics regardless of the planning task. For further examples see Example 23. In our experiments, we frequently observed that the learning identified subtle unexpected dependencies, again suggesting the aforementioned alternative use.

EXAMPLE 23: LEARNED FORMULAS

Let's consider some examples for illustration, covering the different possible relations to the target formula.

(a) $\phi_l = \phi_t$: The same formula is mostly learned for simpler templates like safety, guarantee, a before b or mutual exclusion:

- Transport: $\Diamond(in(p_3, t_0) \wedge in(p_0, t_0))$: At some point packages p_0 and p_3 are together in truck t_0.
- Transport: $\Box\neg(in(p_3, t_1) \wedge in(p_0, t_1))$: Packages p_0 and p_3 are never together in truck t_1.
- Blocksworld: $\neg on(b_3, b_1)$ U $holding(b_5, h_0)$: Hand h_0 holds block b_5 before block b_3 is stacked on block b_1.

(b) ϕ_l is equivalent to ϕ_t: Equivalent formulas can sometimes be identified assuming some basic knowledge about the domain mechanics:

- Parent's Afternoon: $\phi_t = \Diamond(todo(a_2) \wedge \textit{in-car}(p_4))$: At some point person p_4 needs to be in the car and activity a_2 still needs to be done.; $\phi_l = todo(a_2)$ U $\textit{in-car}(p_4)$: Activity a_2 stays undone until person p_4 is in the car. Since doing an activity can not be reversed ϕ_t is equal to the ordering ϕ_l.
- TPP: $\phi_t = \Box(stored(g_1) \rightarrow \Diamond at(t_1, m_2))$: Whenever good g_1 is stored, then truck t_1 visits market m_2 afterwards; $\phi_l = \Diamond(stored(g_1) \wedge at(t_1, m_2))$: At some point good g_1 is stored and truck t_1 is at market m_2. $stored(g_1)$ cannot be reversed and is a goal fact. Thus, there is the equivalent simpler formula ϕ_l.

There are also formulas which are equivalent only in a specific instance:

- TPP: $\phi_t = \neg(loaded(g_2, t_1) \vee at(t_1, m_3))$ U $at(t_1, m_2)$: Truck t_1 does not load good g_2 or visits market m_3 before it visits market m_2; $\phi_l = \neg at(t_1, m_3)$ U $at(t_1, m_2)$: Truck t_1 does not visit market m_3 before it visits market m_2. Good g_2 is offered at market m_2 and m_3. Thus, not visiting m_2 but loading good g_2 Π-entails visiting market m_3.

(c) $\Pi \models \phi_l \rightarrow \phi_t$: Over-approximations can be simple sub-formulas which are relatively easy to identify:

- Transport: $\phi_t = \Box(\neg \textit{at}(p_3, l_1) \vee \textit{at}(p_0, l_4))$: Package p_0 has to be at location l_4 or package p_3 is not at location l_1; $\phi_l = \Box\neg \textit{at}(p_3, l_1)$: Package p_3 is never at location l_1.

But they can also reflect behavior entailed due the underlying planning task:

- Blocksworld: $\phi_t = \Diamond\neg \textit{handempty}(h_0)$: At some point hand h_0 is not empty; $\phi_l = \Diamond \textit{holding}(b_1, h_0)$: At some point hand h_0 holds block b_1. This is an over-approximation, because hand h_0 can hold any block to not be empty.

The entailment is often not only based on the domain mechanics but on the specific instance.

- Transport: $\phi_t = \Box(\neg \textit{at}(p_3, l_1) \vee \textit{at}(p_0, l_4))$: Package p_0 has to be at location l_4 or package p_3 is not at location l_1; $\phi_l = \neg \textit{at}(p_3, l_1)$ U $\Box \textit{at}(p_0, l_4)$: Package p_3 is not at location l_1 until package p_0 is forever at location l_4. This is an over-approximation, because $\textit{at}(p_0, l_4)$ is a goal fact.

(d) $\Pi \models \phi_t \rightarrow \phi_l$: Under-approximations can be simple sub-formulas which are relatively easy to identify:

- Transport: $\phi_t = \neg in(p_4, t_1)$ U $(at(p_1, l_3) \wedge in(p_2, t_0))$: Package p_1 has to be at location l_3 and package p_2 in truck t_0 before package p_4 is in truck t_1; $\phi_l = \Diamond at(p_1, l_3)$: At some point package $p1$ has to be at location l_3, or

$\phi'_l = \neg in(p_4, t_1)$ U $at(p_1, l_3)$: Package $p1$ has to be at location l_3 before package p_4 is in truck t_1.

But they can also reflect behavior entailed due the underlying planning task mechanics or even instance:

- Rovers T: $\phi_t = \neg$*in-memory*(i_1) U *uploaded*(i_0): image i_0 has to be uploaded before image i_1 is taken; $\phi_l = \neg$*uploaded*(i_1) U *uploaded*(i_0): image i_0 has to be uploaded before image i_1 is uploaded. This is an under-approximation, because an image has to be taken before it can be uploaded.
- Blocksworld: $\phi_t = \Box($*on*$(b_4, b_2) \rightarrow ($*on*(b_4, b_2) W $\Box\neg$*on*$(b_4, b_2)))$: Put block b_4 only once on block b_2; $\phi_l = \Box\neg$*on*(b_4, b_2) R $\neg$*clear*(b_1): Unstacking b_4 from b_2 allows clearing b_1. This is an under-approximation, because b_4 is initially on b_2 and b_2 on b_1. b_4 must be unstacked to satisfy all goals and since it cannot be placed again on b_2 it allows clearing b_1 by unstacking b_2. This is an example where the learned formula is harder to understand, although it is smaller than the target formula.

(e) no direct relation, i. e., none of (1)–(4) holds: Especially for smaller number of plans there are often small templates that lead to perfect formulas that are not related to the target formula. Some often appearing templates are:

- Blocksworld: $\Diamond$*holding*(b_2, h_1): Eventually hold block b_2 in hand h_1; $\neg$*holding*(b_4, h_1) U *holding*(b_0, h_1): Hold block b_0 before block b_4 with hand h_1. $\bigcirc$*handempty*(h_1): Start with hand h_0. Many of the formulas describe which of the hands are used for which block.
- Transport: $\Box\neg$*at*(t_0, l_1): Truck t_0 never visits l_1; $\bigcirc$*at*(t_1, l_1): Start by driving with truck t_1 to l_1; $\neg$*at*(p_1, l_3) U *in*(p_0, t_1): Package p_1 stays at l_3 until package p_0 is loaded into truck t_1. Many of the formulas describe which truck is used or in which order the packages are loaded.
- Parent's Afternoon: $\Diamond($*at*(p_2, l_2) R *at*$(p_0, l_0))$: Person p_2 visits l_2 before person p_0 leaves location l_0; $\bigcirc$*at*$(parent, l_3)$: The parent starts with driving to l_3; *done*(a_3) R $\neg$*done*(a_1): Activity a_3 is done before activity a_1. Many of the formulas restrict the order of the activities directly or by restricting the visited locations.

6.3 EMPIRICAL EVALUATION

IMPLEMENTATION

For plan generation, we used the publicly available implementations of *SymK*[1] [Speck et al., 2020] for `TopK` and `TopKFil`, *forbiditerative*[2] [Katz and Sohrabi, 2020a] for `AgDiv`, and *Fast Downward*[3] [Helmert, 2006] for `RNDhFF`. The plan selection for `TopKFil` is supported by *SymK* as an internal filter. We extended the translator of each planner by the LTL_f compilation implementation introduced in Section 4.3.2. The extended planners are publicly available[4]. This is a prerequisite for our experimental setup, as outlined in the next section.

The preferences we consider are LTL_f formulas over facts. To convert plans to lists of fact sets we use VAL [Howey et al., 2004]. We discard the static facts, which are always true, e. g. defining the road connections in transportation domains.

The implementation we use for LTL_f learning is a re-implementation of Camacho and McIlraith [2019]. The original implementation only outputs one formula with the minimum size. As we do not have a reason to prefer one formula over another, we extended the implementation to provide all formulas with the minimum size. This is achieved by repeatedly calling the SAT solver, adding a new clause each time to enforce that previously found formulas are excluded. Our re-implementation is publicly available[5].

6.3.1 EXPERIMENTS SETUP & BENCHMARKS

BENCHMARKS

We consider the instances from the resource and time constrained domains introduced in Section 4.3.2. To allow for a diverse set of plans, we removed all resource and time constraints. Since our experiment setup requires us to solve every task multiple times we only consider the smaller instances.

To generate hidden target formulas for our experiments, we used the LTL_f templates given in Table 14. For each planning task $\tau = (V, A, c, I, G)$ we instantiated each template with random facts from $\bigcup_{a \in A} \mathit{eff}_a$. For templates up to size 5, we also included extended versions, by instantiating a or b with a conjunction or disjunction of two potentially negated facts. Then, for each candidate formula ϕ, we checked whether ϕ is non-tautological: (τ, ϕ) is added to our benchmark set only if both $G \cup \{\phi\}$ and $G \cup \{\neg\phi\}$ were solvable. For each task τ, we included at most three formulas based on the same template. To ensure termination, we skipped a template after at most 5 failed candidate formula checks. This procedure generated an average of 55 formulas per task, resulting in a benchmark set of 5247 task-formula pairs. The benchmark is publicly available[6].

[1] `https://github.com/speckdavid/symk`
[2] `https://github.com/IBM/forbiditerative`
[3] `https://github.com/aibasel/downward`
[4] `https://doi.org/10.5281/zenodo.14989440`
[5] `https://doi.org/10.5281/zenodo.14989191`
[6] `https://doi.org/10.5281/zenodo.14988342`

HYPOTHETICAL BEST-CASE FOR PLAN GENERATION

In practice, plan generation cannot be tailored to the planning formula ϕ_t, as it is hidden in the user's head. Yet, intuitively, it is important for the example plans to be *balanced*: the same numbers of positive and negative instances. A highly imbalanced set of examples can be expected to impede formula learning, because it fails to clarify the distinction between the two classes.

To evaluate this hypothesis, we explored two distinct plan generation setups: the realistic *application setup* Gen_{app} where plans are generated without knowledge of ϕ_t; vs. the hypothetical *idealized setup* $\text{Gen}_{\text{ideal}}$ where we generate perfectly balanced example plan sets by enforcing ϕ_t and $\neg\phi_t$ each in half of the plan generation runs. Kim et al. [2019]'s experiment setup features the same idealized setup, to guarantee the existence of positive and negative sample traces. The idealized setup also sheds light on what could potentially be achieved in future work by advanced methods trying to incorporate partial information about the user preference i. e., what kinds of preference templates or objects are of interest.

EVALUATION WITH RESPECT TO THE TARGET FORMULA

In our experiments, to evaluate the quality of the learned formula, our main criterion is the degree of direct relation to the hidden target formula ϕ_t, according to the categories (a)–(e) discussed in Definition 52. Note that checking these relations involves expensive implication tests to identify entailments in plan space. Our implementation works as follows. The context of each test is a planning task τ with goals G. Given the target formula ϕ_t and the learned formula ϕ_l, we test whether (1) $\Pi \models \phi_l \rightarrow \phi_t$ and (2) $\Pi \models \phi_t \rightarrow \phi_l$. Each test is performed through compilation into a modified planning task, namely $G \cup \{\phi_l, \neg\phi_t\}$ for (1) and $G \cup \{\neg\phi_l, \phi_t\}$ for (2), where LTL_f goals are encoded through compilation into goal facts. Each test succeeds iff the corresponding planning task is unsolvable.

All experiments were run on `Intel E5-2695 v4 2.1G` machines with a memory limit of 4 GB. The example plan generation and the formula learning had time limits of 30min each. As evaluating ϕ_t with respect to ϕ_l can be very time-consuming, we used a timeout of 2 hours for this step. For Gen_{app}, we generated up to 50 example plans, and for $\text{Gen}_{\text{ideal}}$, we generated up to 25 positive and 25 negative examples.

6.3.2 EXPERIMENTAL RESULTS

Our evaluation is divided into two parts. The first part evaluates the plan generation, in terms of computational performance, similarity measures and the balance of the resulting plan sets. The second part evaluates the quality of the learned formulas relative to the hidden target formula, as a function of target formula size, plan generation method, and the number n of annotated example plans.

PLAN GENERATION

Computational Performance Figure 45 shows the average number of plans generated over time. Top-k planning (`TopK`) and randomized h^{FF} (`RNDhFF`) compute all 50 plans

within the first second. Agile diverse planning (AgDiv) generates the requested 50 plans within about 10 sec. Top-k planning with a permutation filter (TopKFil) is least suitable for generating a larger number of sample plans. The combination of top-k search output and permutation filtering proves to be so restrictive that the requested number of plans is not reached within the time limit.

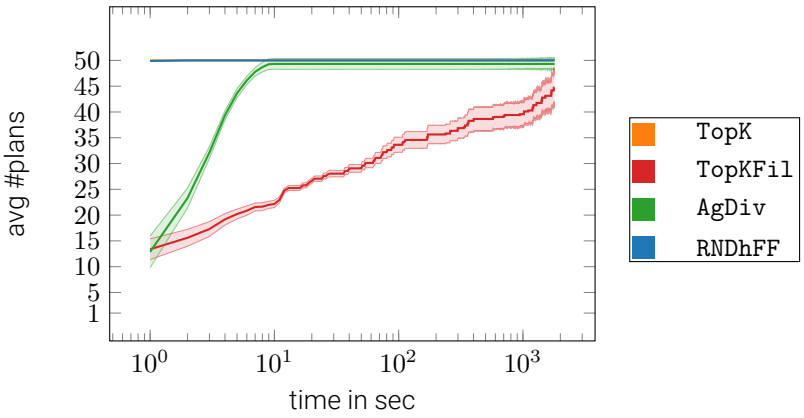


Figure 45: Average number of plans generated over time for each plan generation approach in the application setup Gen_{app}.

Similarity Measures The number of plans and their similarity to each other can influence the number of different ways of solving the task they cover. Therefore, the next step is to evaluate the similarity of the resulting sets of plans. To do so, we use the similarity measures defined by Nguyen et al. [2012] and Katz and Sohrabi [2020b], and compute the average similarity over all pairs of plans in Π. Given two plans π, π', where $A(\pi)$ is the set of actions of plan π we have:

- Uniqueness Similarity: $\delta_u(\pi, \pi') := [A(\pi) \subseteq A(\pi')] + [A(\pi') \subseteq A(\pi)]$ where $[S \subseteq S']$ is 1 if $S \subseteq S'$ and 0 otherwise.
- Stability Similarity: $\delta_a(\pi, \pi') := \frac{|A(\pi) \cap A(\pi')|}{|A(\pi) \cup A(\pi')|}$
- State Similarity: Let $(s_0, s_1, \cdots, s_k)$ and $(s'_0, s'_1, \cdots, s'_k)$ be the set of states traversed by π, π'. Assuming $k' \leq k$ and $\Delta(s, s') := \frac{|s \cap s'|}{|s \cup s'|}$ we have: $\delta_s(\pi, \pi') := \frac{1}{k} \sum_{i=1}^{k'} \Delta(s_i, s'_i)$

The average plan similarity for all three similarity measures is given in Table 15.

As we are measuring similarity and are seeking diverse plans, lower scores are preferable. Overall, RNDhFF generates the most diverse plans, followed by AgDiv and TopKFil, which perform similarly, as expected given their similar filter criteria. TopK performs the worst overall, generating a high number of permutations. On average more than 75% of actions and more than 80% of the states are identical. This suggests that in most domains, there is no large variety among the first 50 optimal plans. RNDhFF and also TopKFil have almost a perfect uniqueness similarity score. With AgDiv and TopKFil on average two plans have about half the actions in common and two thirds of the states. RNDhFF generates

		Blocks	Nomy	Parent's A	RoversR	RoversT	Satellite	TPP	all
δ_u	TopK	0.11	0.8	0.23	1.0	0.11	0.04	0.72	0.4
	TopKFil	0.01	0.02	0.0	0.0	0.01	0.01	0.0	0.01
	AgDiv	0.07	0.02	0.01	0.04	0.16	0.01	0.01	0.05
	RNDhFF	0.0	0.0	0.01	0.0	0.02	0.0	0.0	0.01
δ_a	TopK	0.48	0.95	0.89	1.0	0.75	0.49	0.95	0.78
	TopKFil	0.35	0.49	0.68	0.38	0.52	0.45	0.38	0.47
	AgDiv	0.37	0.53	0.6	0.49	0.6	0.39	0.31	0.48
	RNDhFF	0.26	0.32	0.59	0.26	0.48	0.28	0.41	0.38
δ_s	TopK	0.7	0.81	0.91	0.93	0.85	0.7	0.89	0.82
	TopKFil	0.64	0.47	0.78	0.83	0.62	0.69	0.65	0.67
	AgDiv	0.64	0.49	0.71	0.83	0.6	0.67	0.46	0.63
	RNDhFF	0.4	0.24	0.59	0.74	0.24	0.55	0.28	0.43

Table 15: Average similarity per domain of all plans found per instance for each plan generation approach, rounded to two decimal places.

plans with 10% less stability similarity and 20% less state similarity, compared to AgDiv and TopKFil.

Balance of Plan Sets For the learning step, at least one positive and one negative example is necessary. This is the case for 11% for TopK, 40% for TopKFil, 54% for AgDiv, and 74% for RNDhFF out of the benchmark instances (task-formula pairs). TopK covers only about a tenth of the sample formulas. This is due to its tendency to generate plan permutations. Although, TopKFil and AgDiv have similar similarity scores, AgDiv covers 14% more formulas. RNDhFF, with the lowest similarity score, also covers the largest number of formulas. This suggests that the known similarity measures can serve as a point of reference to determine whether a set of plans is likely to contain positive and negative examples for a variety of temporal preferences.

In what follows, we consider, for each plan-generation method, only those benchmark instances where both positive and negative example plans were generated. The set of all these benchmark instances (union across all plan-generation methods) is denoted by $I_{p\&n}$. Figure 46 evaluates how balanced the sets of example plans are for each plan-generation method.

It may seem surprising that TopK generates the most balanced example plan sets within $I_{p\&n}$, given the results from above. As the red bars in Figure 46 show, TopKFil and AgDiv also tend to be more balanced on these benchmark instances. RNDhFF results in a relatively equal distribution. AgDiv suffers from an uneven behavior, delivering either a very balanced or a very unbalanced set of sample plans. Overall, the superior plan generation algorithm in terms of balancedness is TopKFil.

QUALITY OF LEARNED FORMULAS

Figure 47 and Figure 48 provide our evaluation of the learning quality. Our main criterion for assessing quality are categories (a) - (e) from Definition 52. An instance is considered *solved* if either the target formula or an equivalent formula has been learned.

We consider first Figure 47, which provides data for the realistic plan-generation setup Gen_{app} where the hidden target formula is not taken into account for plan generation. To

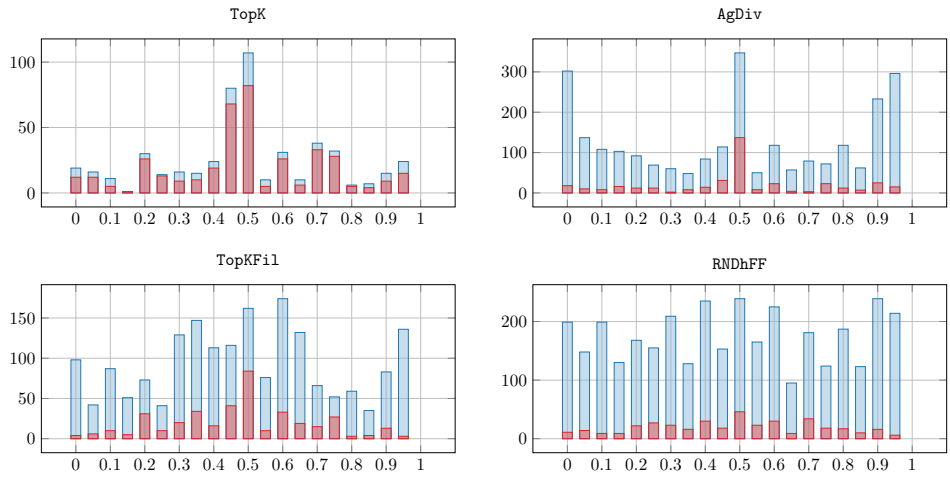


Figure 46: Histogram over relative number of negative examples for each plan-generation method. Bins are half open, value x refers to bin $[x, x + 0.05)$. Blue: over all benchmark instances $I_{p\&n}$; Red: instances for which all plan generation approaches provided at least one positive and one negative example.

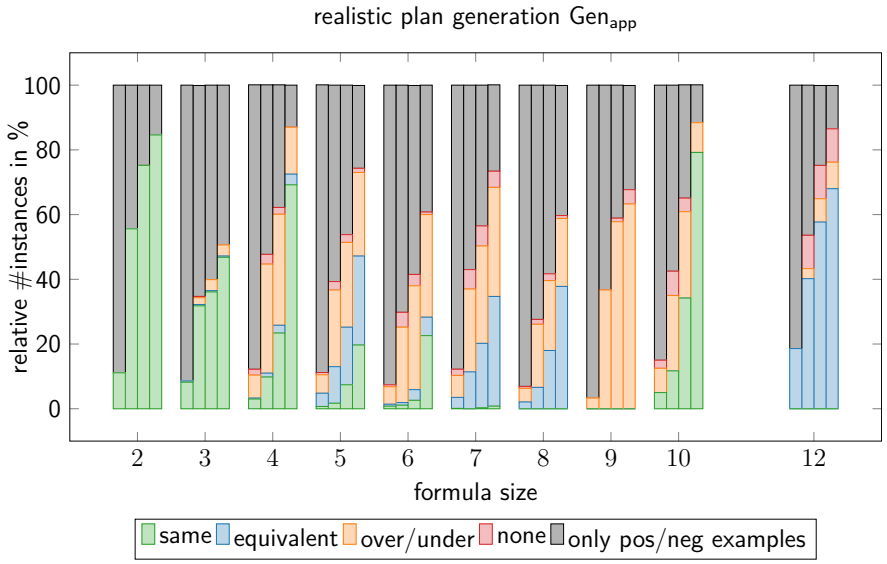


Figure 47: Realistic plan generation: Relative number of instances where the same, an equivalent formula, an over/under-approximation, or no related formula at all is found. Order of plan-generation approaches for each formula size: `TopK`, `TopKFil`, `AgDiv`, `RNDhFF`.

make the complete picture visible, we also include those cases where learning was not possible, as only positive/negative example plans were generated. For very small formulas (size 2 and 3), all plan generation approaches solve almost all the benchmark instances ($I_{p\&n}$) where learning could be run. For larger target formulas, performance declines significantly. Comparing across plan-generation methods `RNDhFF` performs best, followed by `AgDiv`. This ranking of plan generation approaches is exactly the ranking according to the number of instances in $I_{p\&n}$. While `TopKFil` tends to produce more balanced plan

sets than RNDhFF (see Figure 46), it does not produce a better ratio of high-quality learned formulas. For instance, RNDhFF solves significantly more instances for formulas of size 4, suggesting that the diversity of plans has a greater impact on the quality of the learned formula than balanced example sets.

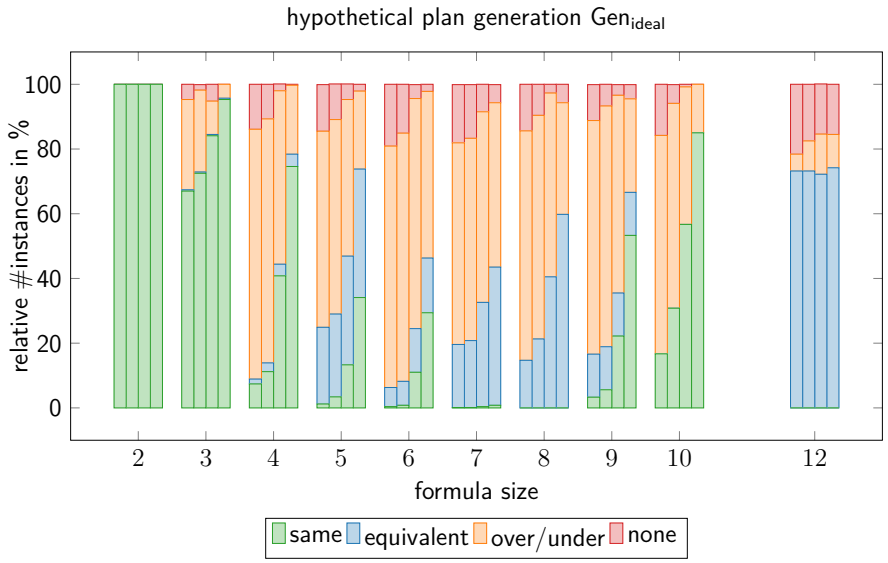


Figure 48: Hypothetical plan generation : Relative number of instances where the same, an equivalent formula, an over/under-approximation, or no related formula at all is found. Order of plan-generation approaches for each formula size: TopK, TopKFil, AgDiv, RNDhFF.

Turning now to Figure 48, we see that the bottleneck of our approach is the quality of plan generation. Recall that in the idealized hypothetical setting Gen$_{\text{ideal}}$, the plan generation methods have access to the target formula and produce balanced example plan sets. The quality of the learned-formulas increases dramatically compared to Gen$_{\text{app}}$. For formula sizes 1 all and 2 about two-thirds are consistently solved, equivalent formulas are often learned even for large formulas, and formulas unrelated to the target are almost never learned. The quality of the learned formulas can differ significantly between the plan generation approaches. RNDhFF provides the best quality, followed by AgDiv. The key question for future work is how to reduce this performance gap between Gen$_{\text{app}}$ and Gen$_{\text{ideal}}$. We return to this in the future work section.

Finally, consider Figure 49, which provides an evaluation of plan generation methods in terms of the number n of example plans required to solve a benchmark instance. To make a meaningful comparison, we need commonly solved benchmark instances, we need to exclude TopK, and exclude uninteresting instances solved by any method with $n = 2$, i. e. one positive and one negative example. Given these restrictions, Gen$_{\text{app}}$ does not provide a sufficient basis for a meaningful comparison, so we consider Gen$_{\text{ideal}}$ instead. Within this selection, shown on the left side of Figure 49, TopKFil performs the best overall. Its median is never greater than $n = 4$, and its variance is small. AgDiv requires on average twice as many examples and RNDhFF three times as many. Looking only at the common instances solved between AgDiv and RNDhFF on the right side of Figure 49 reveals, that in this direct comparison RNDhFF needs fewer sample plans than AgDiv. This shows, that

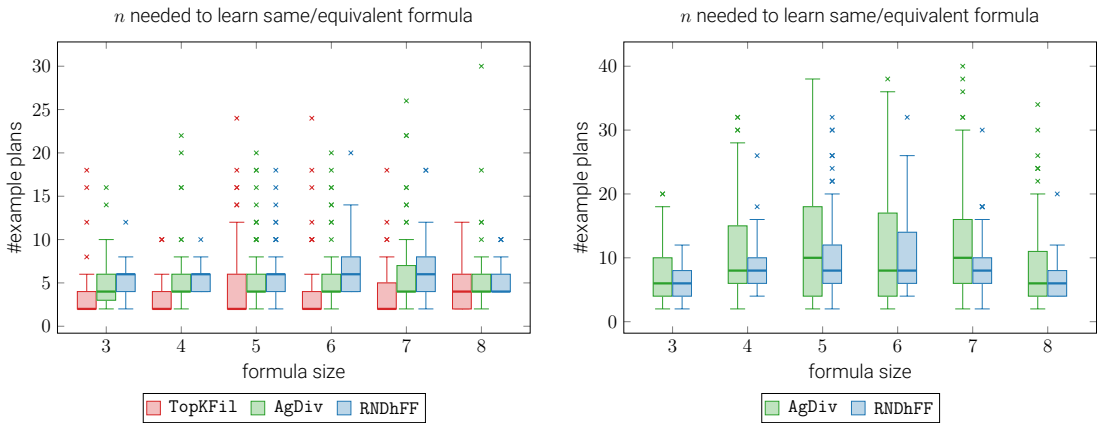


Figure 49: Distribution of number of plans needed, in $\text{Gen}_{\text{ideal}}$ setup, to solve an instance. (left) 345 instances commonly solved by `TopKFil`, `RNDhFF`, and `AgDiv`; (right) 1119 instances commonly solved by `RNDhFF`, and `AgDiv`. Instances solved with 2 plans by all plan generation approaches are not included.

overall `RNDhFF` solves the most instances and requires the least number of plans, with a median that is never larger than 8.

6.4 Discussion

In the following we summarize the contributions and results presented in Chapter 6, discuss related word specific to this chapter, and give an outlook on future work.

Summary

It is often difficult for users to formalize their preferences. Here, we assembled an architecture to learn user preferences from annotated sample plans. We compared different approaches to generate sample plans and evaluated them in terms of diversity and their ability to provide positive and negative examples for a range of commonly used temporal templates. The evaluation revealed a large performance gap between practical plan generation (without access to the hidden target formula) and idealized plan generation (with such access). Thus, the main question for future work is how to effectively generate sample plans, that cover a wider variety of potential user preferences. The results are encouraging and represent a first step towards a more in-depth investigation of this form of preference elicitation in planning.

Related Work

Temporal properties are used in planning for various purposes, e.g. as temporal preferences [Gerevini et al., 2009], for the design of plans or as domain-dependent control knowledge to speed up the computation of plans [Bacchus and Kabanza, 2000]. However, not only for plan generation but also for explanations of plans or the model, temporal properties have been used in various ways. One approach is to summarize plans or traces by identifying

common temporal behavior. Lemieux et al. [2015] provide, given a set of traces and an LTL template, a set of instantiations of the template that are satisfied by all traces. For example instances of the template $\Box(a \rightarrow \bigcirc\Diamond b)$ (a always followed by b) can provide information about the joint occurrence of events.

Kim et al. [2019] address contrastive questions using discriminative temporal properties. The question is given as two sets of contrasting sample traces E^+ and E^-, representing examples of a positive and a negative outcome, leading to the question "Why do the traces E^+ succeed while the traces in E^- fail?" Temporal properties that the positive samples have in common, while they are not satisfied by the negative samples, are used to provide information about common behavior/strategies of successful traces.

Sohrabi et al. [2011] address the questions "How happens A?" from a partially defined state. As an answer, a set of facts that need to hold initially and a plan π satisfying A is given. To provide preferred explanations efficiently they exploit explanation preferences, temporal formulas, that include control knowledge and user preferences.

While not yet using temporal properties, Nguyen et al. [2012] also address the setting where a user has not yet formed their preferences. Instead of first identifying the user's preferences and then allowing the user to explore multiple sample plans in an iterative process, they offer a set of sample plans at once, hoping to cover one of the user's preferred plans. If nothing is known about the user's preferences, they use different domain-independent diversity measures, to generate a diverse set of sample plans. If the users preferences are partially known, i. e. the features of interest (e. g. makespan or cost) and a distribution of weights of their relative importance, diverse plans are generated with respect to those measures.

Cruse and Muise [2022] proposed a procedure that learns a set of temporal formulas that group a set of traces into k clusters. This is done in an iterative procedure, splitting one set at a time. k is automatically determined by maximizing the information gain, i. e. how evenly the sets are split. This approach may be useful for non-binary properties. For example, if there are three rovers that are all equipped to perform a certain task, while none of the rovers is preferred by the user, they would like to analyze the conflicts between the use of each rover. Then three temporal goals reflecting the use of each rover would be suitable.

FUTURE WORK

There are many ways to improve each of the three steps of the preference elucidation process.

Meta-Operators The size of an LTL_f formula does not necessarily correlate with the complexity of its meaning to a human. For example, the template $\Box(a \rightarrow (a \,\mathsf{W}\, \Box\neg a))$ has the simple meaning of *at most once*. The approach by Camacho and McIlraith [2019] does not require templates and finds the smallest formulas with perfect accuracy. However, there may be larger formulas with a simpler meaning that achieve the same accuracy. To keep the freedom to explore formulas that are not covered by any template, while also considering larger formulas with simple meaning, one can use the templates as *meta-operators*. The

size associated with such a meta-operator can be used to rank them according to complexity or user preference.

Partial Plans The number of plans the user has to process before finding the expected preference should be small. In addition to the number of plans, the length of the plans themselves also affects the user's load. For example, if the user's target formula includes the order in which samples are collected, then any actions involving the rovers driving or uploading data are not relevant. Thus, the ability to consider only partial plans could reduce the load on the user. The decision of which actions are relevant could be based on user input. An automatic partitioning of a plan could be based on the decomposition of partial order plans [Young et al., 1994] or plan options as described by Lindsay and Petrick [2021a].

Guided Plan Generation In general, any information provided by the user to narrow down and shape the set of example plan E can be useful. Assuming that the iterative planning process and the identification of new relevant preferences interleave, the soft goals that the user decided to enforce could be used to constrain the plan space for the sample plans.

LTL_f templates can be useful not only in the learning step, but also for the plan generation. Knowing what kind of temporal structures the user is interested in, it may be possible to develop plan-diversity measures tailored to the user's interest.

Noisy Data So far, we have assumed that the user annotates all plans correctly, thus a perfect formula represents what the user had in mind. However, if the user makes a mistake, then the learned formula will capture that mistake. One could follow the approach of Kim et al. [2019], which is based on probabilistic Bayesian models. The approach is robust to noise (wrong annotated examples) because it does not try to find a perfect formula, but instead optimizes accuracy. If you want to remain independent of LTL templates and learn arbitrary formulas then the approach of Gaglione et al. [2021] could be applied.

Workflow for Plan Annotation To facilitate the annotation process, it is essential to implement a suitable interface. Different workflows can be considered. For example, one plan at the time could be presented to the user. The user then has the option to annotate it either as `positive`, `negative` or `neutral`. `Neutral` in this case means, that the plan does not exhibit the user's preference, but also does not contradict it. However, without an alternative, it may be difficult for the user to decide whether they like the plan. So, an alternative approach could be to show two plans at the time and let the user decide, which plan is *better*. This ranking of plans then needs to be transformed into labels by deciding, where to divide the plans into positive and negative examples.

For this approach, plans need to be compared. To facilitate this process, a visualization of the differences between the plans would be useful. Krarup et al. [2021] provide a simple coloring approach, to highlight the differences between an original plan and an alternative plan in their contrastive explanation framework. Chakraborti et al. [2024] implemented a plan selection tool. Based on disjunctive action landmarks, they visualize the choice points in a set of plans.

More Explanations While learned formulas that are not identical to the target formula may not be easily recognizable to the user, as mentioned above, they could also serve as plan space explications. In this alternative application setting, the task is not to learn a hidden target formula, but instead to automatically identify new formulas entailing, or entailed by, a known (previously already specified) plan preference ϕ_t. This may reveal non-obvious properties of the plan space. How these explanations can be incorporated into the iterative planning process remains to be decided.

CHAPTER 7

CONCLUSION

In this work, we sought to address the challenge of developing contrastive explanations for oversubscription planning (OSP). Our objective was to provide explanations that elucidate the dependencies between user preferences and goals, thereby enabling the user to find better trade-offs in an iterative planning process. To this end, we addressed the following three questions:

How can goal conflicts be identified and used to provide contrastive explanations? We have developed an explanation framework based on goal conflicts, answering questions of the form "Why is p not satisfied in the sample plan?" by "Because to satisfy p you have to forgo q which is currently satisfied by the plan." This way, users can obtain information about the conflicts between the goals and their preferences while exploring the plan space. A large-scale online user study demonstrated that the proposed explanations tend to enable users to identify better trade-offs. Two algorithmic approaches have been introduced to compute the minimal unsolvable goal subsets (MUGS) underlying our explanations. We exploited symbolic search for efficient satisfiability checks in a goal space search. In a branch-and-bound approach, a heuristic used for pruning was introduced to estimate whether the set of maximal solvable goal subsets can be further improved. This heuristic was instantiated based on admissible heuristics for individual goals. The result was a computational performance comparable to that required to compute an optimal plan for an OSP task.

Once a conflict has been identified, a natural follow-up question is: *Why does a goal conflict exist, and how can it be resolved?* Conflicts are often caused by constraints, such as limited resources or time. Minimal relaxations of these constraints that resolve the conflict provide a natural explanation, offering the user information on the underlying causes of the conflict, as well as on how it can be resolved. To this end, we have expanded our framework to include explanations that take up this idea. Two algorithmic approaches have been introduced to compute the MUGS for a given set of relaxations by exploiting information that can be propagated between relaxed tasks, such as reachable goals and the search space. Furthermore, we explored a compilation approach based on relaxations expressed as temporal soft goals. The empirical evaluation demonstrated that relaxation explanations

can be computed for a reasonable number of relaxations.

Valuable explanations must cover aspects of the plans in which the user is interested. However, it is often challenging for users to formalize their preferences. Therefore, we addressed the question: *How to find suitable properties to provide valuable explanations ?* We compiled and evaluated a framework for learning user preferences from annotated sample plans. The results are encouraging and represent a first step towards a more in-depth investigation of this form of preference elicitation in planning. The evaluation revealed that current approaches to generating diverse plans are not yet suitable for providing plans that reflect a wide variety of possible user preferences. This prompts the question of plan diversity measures that are tailored to partially known user preferences.

The importance of AI systems that can provide explanations for their decisions is growing. To successfully apply explanations based on goal conflicts to real-world problems and to support the interaction with laypeople, some critical points must be addressed. For larger tasks, it is unlikely that all MUGS can be computed, necessitating the use of approximations. Methods that prioritize the exploration of the most significant conflicts using learned knowledge could be a viable option. While classical planning allows for the modeling of a wide range of problems, an extension of goal conflict explanations to more expressive planning formalisms that encode numerical variables, time, and probabilities would be of significant value. This presents a challenge in terms of efficiently computing conflicts between goals/properties, as well as defining them and utilizing them effectively as explanations. As the fields of planning and learning become increasingly intertwined, it becomes imperative to explain not only the plan space but also the method used to generate the plans.

Providing explanations to a human user is a two-step process [Miller, 2019]. First, the causal information for the explanations must be identified, and then the relevant parts must be communicated to the user appropriately. In this work, our focus was on the first step. The second step is equally relevant for a successful explanation and may involve areas such as data visualization, natural language processing, and knowledge engineering. This underlines the multidisciplinary nature of XAI and the major challenges that must be overcome in the future to ensure safe, trustworthy, satisfying and successful use of AI systems.

APPENDIX A

APPENDIX CHAPTER 2

A.1 COMPUTATIONAL EVALUATION: DATA FOR INDIVIDUAL DOMAINS

As illustrated in Figures 50 he average number of MUGS per domain varies significantly across different cost bounds. Some domains have more than 1000 MUGS. The specific trends observed in domains with cost bounds of 0.25, 0.5, and 0.75 are dependent on the domain. For 19 domains the maximum number of MUGS is achieved at at a cost bound of 0.5, while for 17 domains it continuously decreases, and for 9 the number of MUGS increases. As the cost bound increases, the size of the MUGS increases as well. Figure 51 illustrated this relationship. For the tightest cost bound, the MUGS are for most domains not larger than 3. For larger cost bounds, the size increases up to 3 or 4, rarely exceeding 6.

Figure 52 compares the fraction of the goal lattice explored by systematic weakening (SysW) and strengthening (SysS) for each domain. One can clearly see how the explored fraction decreases or increases with increasing cost bound for SysW and SyS, respectively. In 28 out of 45 domains, the explored goal lattice fraction is larger for SysS than for SysW when the cost bound is 0.75. For the smaller cost bounds SysW explores more than SyS.

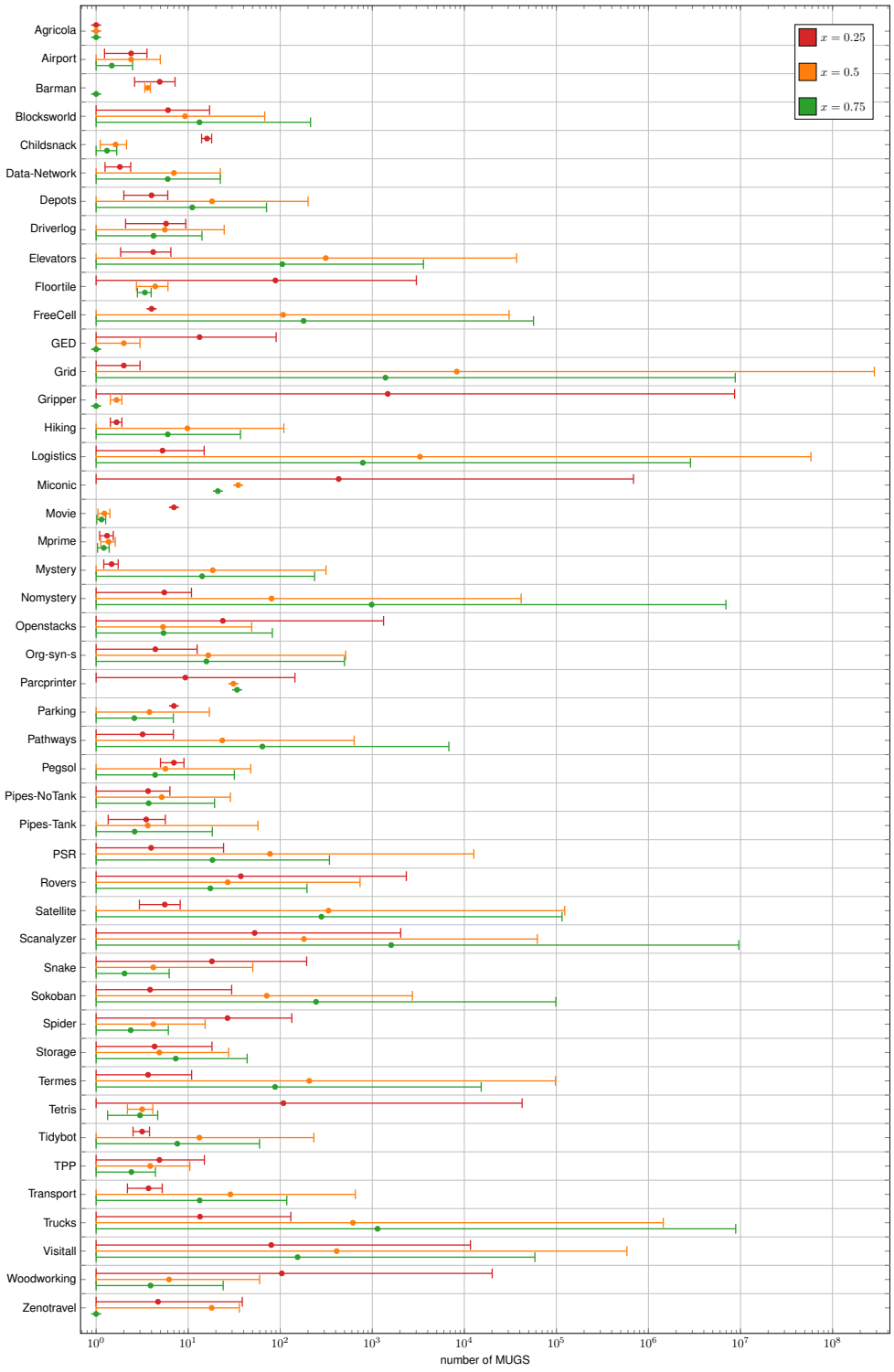


Figure 50: Average number of MUGS per IPC domain over all instances with data for all three cost bounds. Cost bounds are encoded by color, from top to bottom 0.25, 0.5 and 0.75. The center dot is the geometric mean and the error bars represent the variance.

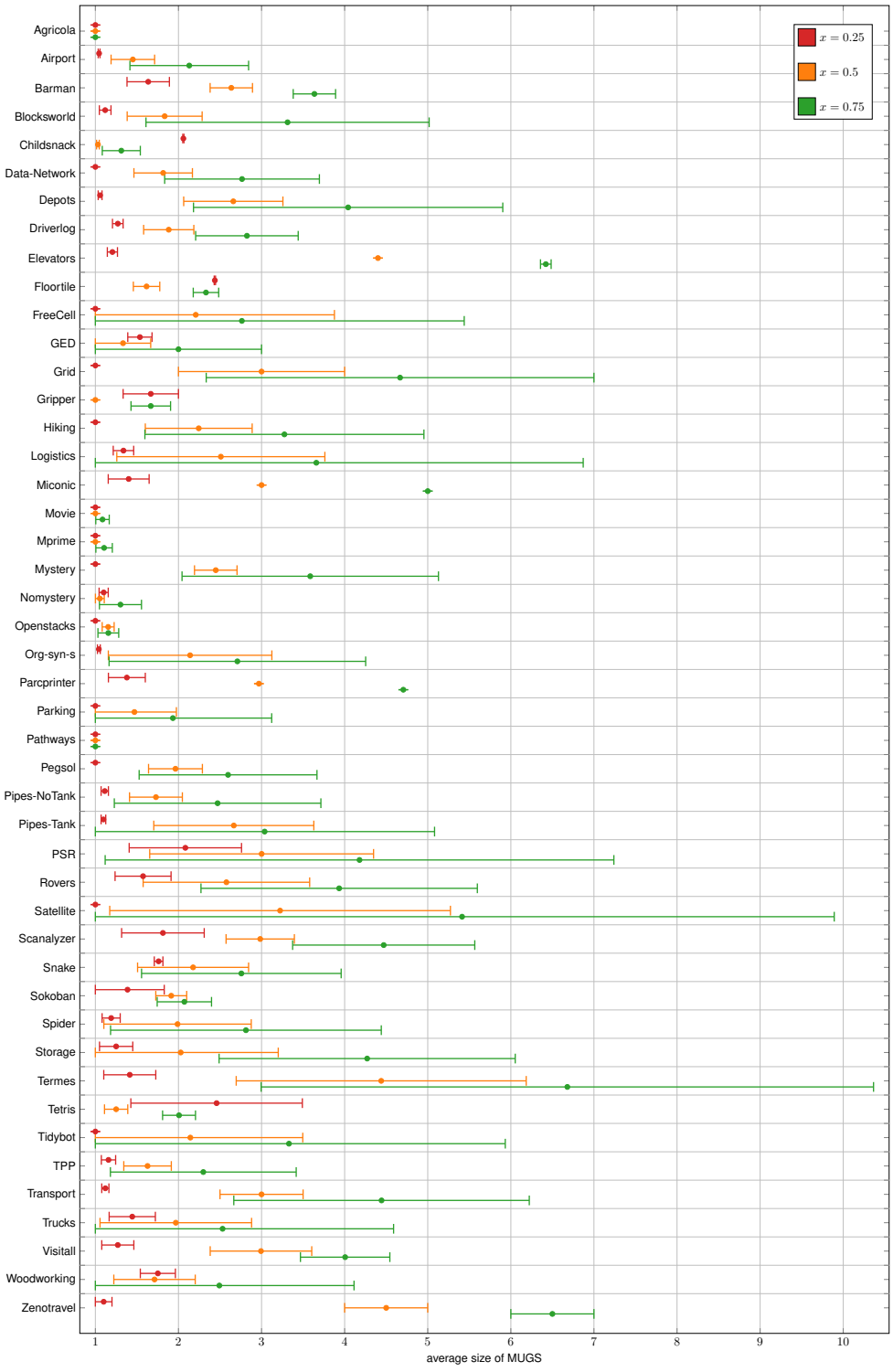


Figure 51: Average size of MUGS per IPC domain over all instances with data for all three cost bounds. Cost bounds are encoded by color, from top to bottom 0.25, 0.5 and 0.75. The center dot is the geometric mean and the error bars represent the variance. Only instances with less than 100 MUGSare considered.

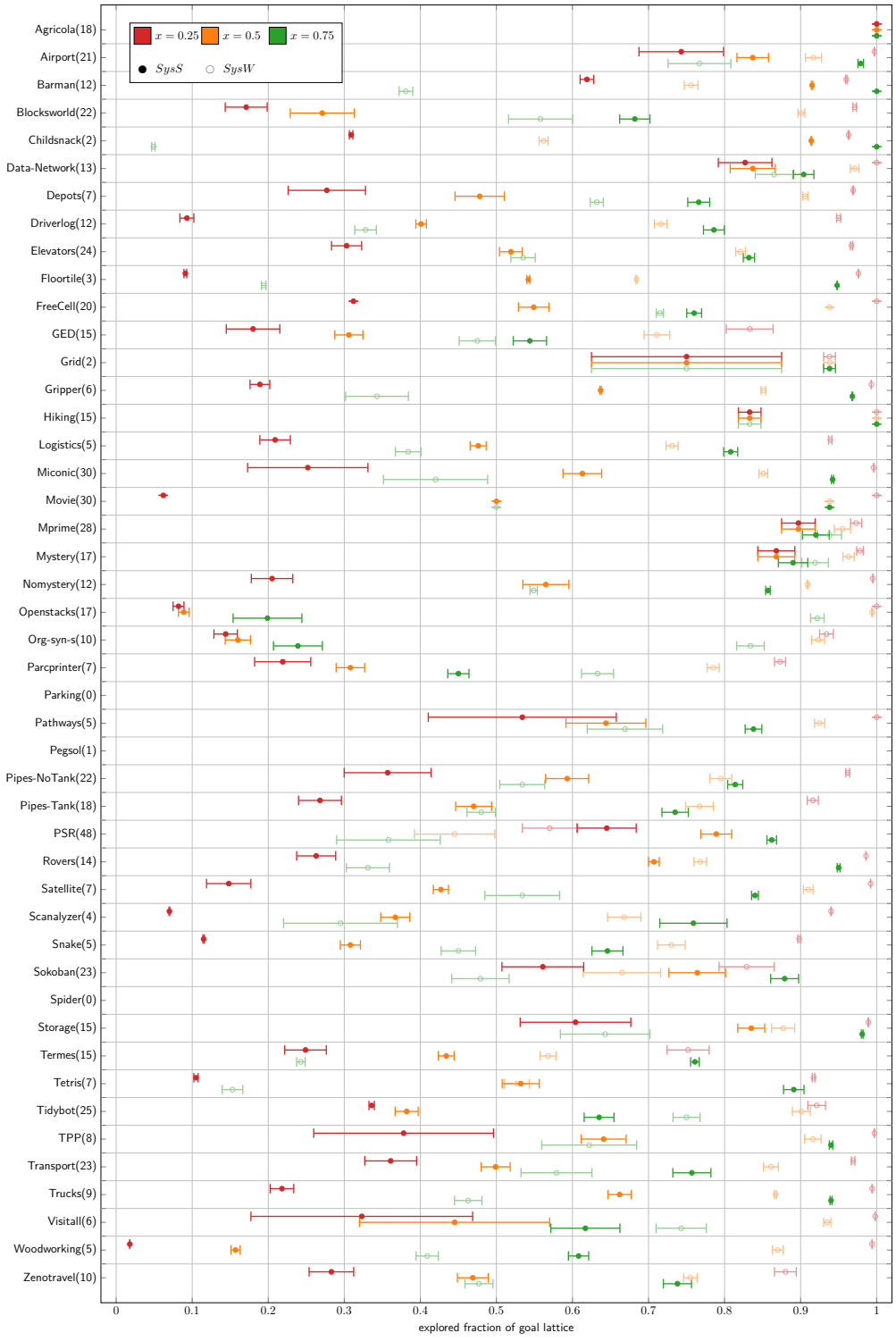


Figure 52: Comparison of average fraction of solved nodes in goal lattice for SysS (strong color) and SysW (light color) per domain. Only instances commonly solved by SysS and SysW for all cost bounds are used. The error bars represent the variance.

A.2 TEMPORAL GOALS: PROOFS AND INPUT DEFINITION

A.2.1 PROOFS

PROPOSITION 26: CORRECTNESS LTL$_F$ COMPILATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, G_{LTL_f} a set of LTL$_f$ temporal goals and $\tau' = (V', A', c', I', G^{\text{hard}'}, G^{\text{soft}'}, b)$ is the LTL$_f$-compilation based on Definition 27.
Let the plan mapping $\beta : A'^* \mapsto A^*$ be defined as $\beta(\pi = a_0 \cdots a_n) = \beta(a_0) \cdots \beta(a_n)$ with

$$\beta(a) = \begin{cases} \epsilon & \text{if } a \in A_N \\ a' \text{ with } pre'_a = pre_a[A] \text{ and } \mathit{eff}'_a = \mathit{eff}_a[A] & \text{otherwise} \end{cases}$$

where $P[V]$ is the projection of partial assignment P to the variables in V.
Task τ' is a temporal goal compilation according to Definition 25.

Proof:
Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and p_t a temporal goal defined in LTL$_f$. Let $N = (S, \Sigma, T, s_0, S_a)$ be a complete NFA accepting traces satisfying p_t and task $\tau' = (V', A' = A_\tau \cup A_N, c, I', G^{\text{hard}'}, G^{\text{soft}'}, b')$ the LTL$_f$ compilation where the soft goal fact g_{p_t} indicating the satisfaction of p_t is given by v_a.
Because all removed actions have cost 0, both plans have the same cost, cost$(\pi) =$ cost$(\beta(\pi))$. Because N is complete there exists a relation $\beta^{-1} : A^* \mapsto \mathcal{P}(A'^*)$:

$$\beta^{-1}(\pi = a_0 \cdots a_n) = \{a_0^N a_0^\tau \cdots a_n^N a_n^\tau a_{n+1}^N \mid a^N \in A[s] \subseteq A_N\}$$

where $a_i^\tau = a$ with $pre_{a_i^\tau} = pre_a \cup \{\mathit{sync} = \tau\}$ and $\mathit{eff}_{a_i^\tau} = \mathit{eff}_a \cup \{\mathit{sync} = N^0\}$. Because all added actions have cost 0, the plan cost stays the same, for all $\pi' \in \beta^{-1}(\pi)$: cost$(\pi) =$ cost(π').

(1) $\forall \pi \in \Pi(\tau) : \pi \vDash p_t \rightarrow \exists \pi' \in \Pi(\tau_{p_t}) : \beta(\pi) = \pi' \wedge g_{p_t} \in I_{p_t}[\![\pi']\!]$:
Let plan $\pi \in \Pi(\tau)$ and $\pi \vDash p_t$. Thus, there exists an execution of N ending in an accepting state. Let E be the set of all accepting executions of N for p_t. For every execution $e \in E$ there exists a plan π' in $\beta^{-1}(\pi)$ with cost$(\pi') < b$ whose automaton actions follow e. This holds, because the automaton actions have the same preconditions as the automation transitions, expect for the *sync* variable, which is satisfied based on the construction in β^{-1} and all added actions have cost 0. Let Π_E be the plans in $\beta^{-1}(\pi)$ whose automaton actions follow an execution in E. From (2) and $\beta(\pi') = \pi$ follows $\Pi_E \subseteq \Pi(\tau_{p_t})$. For all plans $\pi' \in \Pi_E : g_{p_t} \in I_{p_t}[\![\pi]\!]$, because of the definition of Π_E.

(2) $\Pi(\tau) = \{\beta(\pi') \mid \pi' \in \Pi(\tau_{p_t})\}$:

$\subseteq$: Let plan $\pi \in \Pi(\tau)$. $\beta^{-1}(\pi) \subseteq \Pi(\tau_{p_t})$ because, of the definition of β^{-1}, the fact that the added actions from A_N only effect $V' \setminus V$, *sync* is satisfied due to the alternation of task and automata actions in β^{-1} and the cost does not change. Thus, all plan in $\beta^{-1}(\pi)$ are applicable in I' and satisfy the hard goals. Thus, for all $\pi' \in \beta^{-1}(\pi)$ we have $\pi' \in \Pi(\tau_{p_t})$.

$\supseteq$: Let plan $\pi \in \Pi(\tau_{p_t})$. $\beta(\pi) \in \Pi(\tau)$ because all actions removed from π only effect $V' \setminus V$, *sync* which is completely removed, and the action cost stays the same. Thus, $\beta(\pi)$ is applicable in I and satisfies the hard goals.

(3) $\forall \pi \in \Pi(\tau_{p_t}) : G^{\text{soft}} \cap I_{p_t}[\![\pi]\!] = G^{\text{soft}} \cap I[\![\beta(\pi)]\!]$:
Follows the same reasoning as for (2).

□

PROPOSITION 27: CORRECTNESS ACTION-SET COMPILATION

Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, G_{LTL_f} a set of LTL$_f$ temporal goals and $\tau' = (V', A', c', I', G^{\text{hard}'}, G^{\text{soft}'}, b)$ is the Action-Set compilation based on Definition 29.
The plan mapping $\beta : A'^* \mapsto A^*$ is given by $\beta(\pi = a_0 \cdots a_n) = \beta(a_0) \cdots \beta(a_n)$ with

$$\beta(a) = \begin{cases} \epsilon & \text{if } a \in A_{g_\phi} \\ a' \text{ with } pre'_a = pre_a[A] \text{ and } \mathit{eff}'_a = \mathit{eff}_a[A] & \text{otherwise} \end{cases}$$

where $P[V]$ is the projection of partial assignment P to the variables in V.
Task τ' is a temporal goal compilation according to Definition 25.

Proof:
Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task and ϕ a temporal goal defined as Action-Set soft goal. Let task $\tau' = (V', A' = A_\tau \cup \{a_{cp}\} \cup A_{g_\phi}, c, I', G^{\text{hard}'}, G^{\text{soft}'}, b')$ be the Action-Set compilation where the soft goal fact g_{p_t} indicating the satisfaction of ϕ is given by $isTrue_\phi$.
The revers mapping $\beta^{-1} : A^* \mapsto \mathcal{P}(A'^*)$ is given by:

$$\beta^{-1}(\pi = a_0 \cdots a_n) = \{a_0^\tau \cdots a_n^\tau a_{cp} a_e \mid a_e \in A_{g_\phi} \cap A[I[\![a_0^\tau \cdots a_n^\tau a_{cp}]\!]]\}$$

where $a_i^\tau = a$ with $pre_{a_i^\tau} = pre_a \cup \{\mathit{phase} = \mathit{plan}\}$.

(1) $\forall \pi \in \Pi(\tau) : \pi \vDash p_t \rightarrow \exists \pi' \in \Pi(\tau_{p_t}) : \beta(\pi) = \pi' \wedge g_{p_t} \in I_{p_t}[\![\pi']\!]$:
Let plan $\pi \in \Pi(\tau)$ and $\pi \vDash p_t$. Thus, there exists a clause in p_t that is satisfied, which implies that there is an applicable evaluation action in A_{g_ϕ}, that sets $isTrue_\phi = \top$. Let Π_S be the plans in $\beta^{-1}(\pi)$ with the corresponding evaluation actions. From (2), $\beta(\pi') = \pi$ and cost(π') = cost(π) follows $\Pi_S \subseteq \Pi(\tau_{p_t})$. For all plans $\pi' \in \Pi_S$: $g_{p_t} \in I_{p_t}[\![\pi]\!]$, because of the definition of Π_S.

(2) $\Pi(\tau) = \{\beta(\pi') \mid \pi' \in \Pi(\tau_{p_t})\}$:

$\subseteq$: Let plan $\pi \in \Pi(\tau)$. $\beta^{-1}(\pi) \subseteq \Pi(\tau_{p_t})$ because, of the definition of β^{-1}, the fact that the added actions from A_{g_ϕ} only effect $V' \setminus V$, that *phase* is satisfied in the initial state and only changes by the action a_{cp} and that the plan cost stays the same. Thus, all plan in $\beta^{-1}(\pi)$ are applicable in I' and satisfy the hard goals. Thus, for all $\pi' \in \beta^{-1}(\pi)$, with $\beta(\pi') = \pi$ we have $\pi' \in \Pi(\tau_{p_t})$.

$\supseteq$: Let plan $\pi \in \Pi(\tau_{p_t})$. $\beta(\pi) \in \Pi(\tau)$ because all actions removed from π only effect $V' \setminus V$, *phase* is completely removed, and the plan cost stays the same. Thus, $\beta(\pi)$ is applicable in I and satisfies the hard goals.

(3) $\forall \pi \in \Pi(\tau_{p_t}) : G^{\text{soft}} \cap I_{p_t}[\![\pi]\!] = G^{\text{soft}} \cap I[\![\beta(\pi)]\!]$:
Follows the same reasoning as for (2).

□

A.2.2 INPUT DEFINITION

Temporal goals are specified in JSON format and passed to our implementation based on Fast Downward[1] as an additional input. The supported input format is specified in Figure 53.

```
{
    "plan_properties": [temporal_goal]
    "hard_goals": [goal_name]
    "soft_goals": [goal_name]
}


temporal_goal :=
{
    "name": <unique_name>,
    "type": AS|LTL,
    "formula": <formula>,
    "actionSets": [action_set]
}

action_set :=
{
    "name": <unique_name>,
    "actions": [action]
}

action :=
{
    "name": <name>,
    "params": [object|type]
}
```

Figure 53: Supported input format for temporal goals.

The type of a temporal goal, AS or LTL, determines the compilation used following Definition 29 for temporal goals defined by Action-Sets and Definition 27 for LTL_f. In

[1] https://github.com/aibasel/downward

both cases the formulas are defined in prefix notation and can contain predicates (e. g.. `at(t1,l1)`) and names of action sets. Action-Set formulas must be in disjunctive normal form. The action definitions support objects and types as parameters. If only the type is specified, all grounded actions that match the template are included in the action set.

Example definitions are given in Figure 54. The Action-Set goal in the top specifies, that no rover moves to the location `l2`. The action set `move_to_l2` contains all move actions of any rover from any location with any amount of fuel to location `l2`. The LTL_f goal in the bottom specifies, that soil sample `soil_sample0` is uploaded before X-ray image `x_ray_image1`.

```
{
    "name": "never_move_to_locationl2",
    "type": "AS",
    "formula": "! move_to_l2",
    "actionSets":[
        {
            "name": "move_to_l2",
            "actions":[
                {"name": "move", "params": ["rover", "location", "l2", "level
                    ", "level", "level", "level", "level", "level"]}
            ]
        }
    ]
}

{
    "name": "do_uploaded_soil_sample0_before_uploaded_x_ray_image1",
    "type": "LTL",
    "formula": "U ! uploaded(x_ray_image1) uploaded(soil_sample0)"
}
```

Figure 54: Example for Action Set (top) and LTL_f (bottom) temporal goal definition

Templates for Temporal Goals Table 16 lists all templates used in the evaluation of the Action-Set and LTL_f encoding of temporal soft goals. The templates that can be encoded as Action-Set goals are marked with (AS). Section 4.3.2 on generating the benchmark set explains how the templates are used.

These or similar templates can be used in our online iterative planning tool IPEXCO, to simplify the process of temporal goal definition as shown in Figure 23 and described in the next section.

domain	temporal goal template
Blocksworld	Stack block `bx` on block `by` at some time. (AS)
	Do not use hand `h` for block `b`. (AS)
	Put block `b` on the table at some point. (AS)
	Never put block `b` on the table. (AS)
	Hold block `b1` before block `b2`.
	Use hand `hx` before hand `hy`.
	Block `bx` and `by` are both on the table at some point.

Continued on next page

	Block bx and by are never held together.
	Block b is hold at most once.
Nomystery	Use the road between location l0 and l1. (AS)
	Do not use the road between location l0 and l1. (AS)
	Use the same truck for package px and package py. (AS)
	Deliver package p1 before package p1.
	Package px and package py are never together in the same truck.
	Package p is loaded at most once.
Parent's Afternoon	Drive to location l at some point.
	Never drive to location l. (AS)
	Do activity a not at time point t. (AS)
	Do activity a at time point t. (AS)
	Person p1 and person p2 are never together in the car.
	Item ix and item iy are never together in the car.
	Do activity a1 before activity a2.
	Visit location l before location l2.
	Visit location l at most once.
	Person p is in the car at most once.
Rovers R	Use the road between waypoint wx and wy. (AS)
	Do not use the road between waypoint wx and wy. (AS)
	Use rover r to take rock sample s. (AS)
	Use rover r to take soil sample s. (AS)
	Use rover r to take image i. (AS)
	Use camera c to take image i. (AS)
	Upload data dx before data dy.
	Move rover r only once.
Rovers T	Visit waypoint w with rover r. (AS)
	Never visit waypoint w with rover r. (AS)
	Idle at some point. (AS)
	Never idle. (AS)
	Collect sample sx before sample sy.
	Visit waypoint wx with rover r before waypoint wy.
	Sample sx and sample sy never together in memory of rover r.
	Visit waypoint w with rover r at most once.
	Do not visit waypoint w with rover r while sample s in memory.
Satellite	Use satellite s to take image i. (AS)
	Never turn satellite s into direction d. (AS)
	Turn satellite s into direction d at some point. (AS)
	Switch on instrument i at most once.
	Point into direction d at most once.
	Calibrate instrument ix before instrument iy.
TPP	Use the road between location l0 and l1. (AS)
	Do not use the road between location l0 and l1. (AS)

Continued on next page

	Use the same truck for goods gx and goods gy. (AS)
	Load goods gx before goods gy.
	Goods gx and goods gy never together in the same truck.
	Buy goods g at most once.

Table 16: Templates for temporal goals. Temporal goals that can be encoded as Action-Set goal are marked with (AS).

A.3 IPEXCO: Input Definition

In IPEXCO, temporal goals can be defined by directly specifying the LTL_f or Action-Set formula and the necessary action sets, as outlined in Section A.2.2. However, for common temporal goals based on the same pattern, it is more convenient and efficient to define a template. This approach allows the user to simply select the desired instantiation of the template, specifying only the specific objects. An example of such a template is provided in Figure 55.

```
{
    "class": "Road Usage",
    "type": "AS",
    "variables": [
        {
            "name": "$T",
            "type": "truck"
        },
        {
            "name": "$Li",
            "type": "location"
        },
        {
            "name": "$Lj",
            "type": "location"
        }
    ],
    "nameTemplate": "use$Li$Ljwith$T",
    "formulaTemplate": "useConnection",
    "actionSetsTemplates": [
        {
            "name": "useConnection",
            "actionTemplates": [
                "drive $T $Li $Lj fuellevel fuellevel fuellevel",
                "drive $T $Lj $Li fuellevel fuellevel fuellevel"
            ]
        }
    ],
    "sentenceTemplate": "The road between $Li and $Lj is used with Truck $T",
    "initVariableConstraints": ["connected($Li,$Lj)"]
}
```

Figure 55: Example for template definition.

This template is for an Action-Set goal with that states: Truck `$T` has to use the connection between locations `$Li` and `$Lj`. `variables` defines the input parameters of the

template, including their type. `formulaTemplate` and `actionSetsTemplates` follow the same pattern as described in Section A.2.2. The `sentenceTemplate` is used in the interface to select the objects and during the iterative planning process to display the goal. The corresponding interface is shown in Figure 23.

To ensure that only meaningful instantiations are allowed, it is essential to consider more than just the type of input parameter. Additional restrictions may be necessary for certain templates. In our example, using a connection only makes sense if such a connection exists. Therefore, it is possible to define additional constraints in `initVariableConstraints` that must be satisfied by the input parameters in the initial state, e. g. `$Li` and `$Lj` must be connected.

The property `class` is used to group templates with similar meanings in the interface to define new goals.

A.4 Additional Material User Study

In the following we list all goals and the corresponding MUGS used in the three evaluation instances of the user study. In each domain, there was one hard goal, which is marked with GH. Since these goals must be satisfied by all plans, they are not part of the MUGS.

Transport

Goals:

1. Package 0 is delivered to Ms. Lopez. *d*(p_0, l)
2. Package 1 is delivered to Mr. Smith. *d*(p_1, s)
3. Package 2 is delivered to Ms. Jones. *d*(p_2, j)
4. Package 3 is delivered to Mr. Taylor. *d*(p_3, t)
5. Package 4 is delivered to Post Office. (HG)
6. Truck 2 visits the supermarket. $v(sm, t_2)$
7. Package 2 is delivered *before* package 4.*before*(p_2, p_4)
8. The road between the bank and the supermarket and is used by truck 1. *use*(b, sm, t_1)
9. The road between Ms. Lopez' house and teh packing station is used with truck 1. *use*(l, ps, t_1)
10. The road between the post office and the supermarket is not used by truck 1. *not-use*(j, s, t_2)
11. The road between the cafe and Ms. Jones's house and is used by truck 2. *use*(c, j, t_2)
12. The road between Ms. Jones's house and Ms. Smith's house is not used by truck 2. *not-use*(po, sm, t_1)

MUGS:

1. $\{$*d*(p_1, s), *use*(l, ps, t_1), *before*$(p_2, p_4)\}$
2. $\{$*d*(p_1, s), *d*(p_2, j), *d*$(p_3, t)\}$
3. $\{$*d*(p_0, l), *d*(p_3, t), *before*$(p_2, p_4)\}$
4. $\{$*d*(p_0, l), *d*(p_2, j), *d*$(p_3, t)\}$
5. $\{$*d*(p_2, j), *d*(p_3, t), *use*$(b, sm, t_1)\}$

6. $\{d(p_0, l), d(p_1, s), before(p_2, p_4)\}$
7. $\{d(p_1, s), d(p_3, t), before(p_2, p_4)\}$
8. $\{d(p_0, l), d(p_1, s), v(sm, t_2)\}$
9. $\{d(p_0, l), d(p_1, s), use(l, ps, t_1)\}$
10. $\{d(p_0, l), d(p_1, s), use(c, j, t_2)\}$
11. $\{d(p_0, l), d(p_1, s), d(p_3, t)\}$
12. $\{d(p_1, s), d(p_3, t), use(l, ps, t_1)\}$
13. $\{d(p_3, t), use(b, sm, t_1), before(p_2, p_4)\}$
14. $\{d(p_0, l), d(p_1, s), d(p_2, j)\}$
15. $\{d(p_1, s), d(p_2, j), not\text{-}use(po, sm, t_1), use(b, sm, t_1)\}$
16. $\{d(p_0, l), use(l, ps, t_1), v(sm, t_2), before(p_2, p_4)\}$
17. $\{d(p_1, s), use(b, sm, t_1), v(sm, t_2), before(p_2, p_4)\}$
18. $\{d(p_2, j), d(p_3, t), use(c, j, t_2), not\text{-}use(j, s, t_2)\}$
19. $\{d(p_2, j), use(l, ps, t_1), use(b, sm, t_1), v(sm, t_2)\}$
20. $\{d(p_1, s), not\text{-}use(po, sm, t_1), use(b, sm, t_1), before(p_2, p_4)\}$
21. $\{d(p_0, l), d(p_1, s), not\text{-}use(po, sm, t_1), use(b, sm, t_1)\}$
22. $\{d(p_1, s), d(p_2, j), use(l, ps, t_1), use(b, sm, t_1)\}$
23. $\{d(p_1, s), d(p_2, j), use(b, sm, t_1), v(sm, t_2)\}$
24. $\{use(l, ps, t_1), use(b, sm, t_1), v(sm, t_2), before(p_2, p_4)\}$
25. $\{d(p_1, s), not\text{-}use(po, sm, t_1), use(l, ps, t_1), use(b, sm, t_1)\}$
26. $\{d(p_2, j), d(p_3, t), use(c, j, t_2), v(sm, t_2)\}$
27. $\{d(p_3, t), use(c, j, t_2), not\text{-}use(j, s, t_2), before(p_2, p_4)\}$
28. $\{d(p_3, t), use(c, j, t_2), v(sm, t_2), before(p_2, p_4)\}$
29. $\{not\text{-}use(po, sm, t_1), not\text{-}use(j, s, t_2), use(l, ps, t_1), use(b, sm, t_1), before(p_2, p_4)\}$
30. $\{d(p_2, j), not\text{-}use(po, sm, t_1), not\text{-}use(j, s, t_2), use(l, ps, t_1), use(b, sm, t_1)\}$
31. $\{d(p_1, s), use(c, j, t_2), use(l, ps, t_1), use(b, sm, t_1), v(sm, t_2)\}$
32. $\{d(p_1, s), d(p_2, j), use(c, j, t_2), use(l, ps, t_1), v(sm, t_2)\}$
33. $\{d(p_0, l), not\text{-}use(po, sm, t_1), use(b, sm, t_1), v(sm, t_2), before(p_2, p_4)\}$
34. $\{d(p_0, l), d(p_2, j), not\text{-}use(po, sm, t_1), use(l, ps, t_1), use(b, sm, t_1)\}$
35. $\{d(p_0, l), not\text{-}use(po, sm, t_1), use(l, ps, t_1), use(b, sm, t_1), before(p_2, p_4)\}$
36. $\{d(p_0, l), use(c, j, t_2), not\text{-}use(j, s, t_2), use(b, sm, t_1), v(sm, t_2), before(p_2, p_4)\}$
37. $\{d(p_0, l), d(p_3, t), use(c, j, t_2), not\text{-}use(po, sm, t_1), use(l, ps, t_1), use(b, sm, t_1)\}$

PARENT'S AFTERNOON

Goals:

1. Shopping is done. *do*(so)
2. Grandma's shopping is done. *do*(spg)
3. Parent's sports is done. *do*(sp)
4. Soccer training is done. *do*(st)
5. Music Lesson is done. *do*(ml)
6. Bring friend to sport center. *do*(fs)
7. Kid1 is back home. (HG)
8. Kid2 is back home. *bring*(k_2, h)
9. Groceries are at home. *bring*(g, h)
10. Grandma's groceries are at grandma's house. *bring*(g, gm)
11. Grandma is back home. *bring*(gm, gm)
12. Shopping is done *before* sports. *before*(so, sp)
13. Grandma and friend are not together in the car. *not-t*(gm, f)

MUGS:

1. $\{do(ml), before(so, sp)\}$
2. $\{bring(g, gm), before(so, sp)\}$

3. $\{do(st), before(so, sp)\}$
4. $\{do(sp), bring(g, gm), do(st)\}$
5. $\{do(fs), do(st), do(spg)\}$
6. $\{do(sp), do(st), do(spg)\}$
7. $\{do(fs), do(ml), do(st)\}$
8. $\{do(sp), do(ml), do(st)\}$
9. $\{do(fs), bring(g, gm), do(st)\}$
10. $\{do(fs), bring(g, h), do(st)\}$
11. $\{do(fs), do(spg), before(so, sp)\}$
12. $\{bring(g, gm), do(ml), do(st)\}$
13. $\{do(sp), bring(g, h), do(st)\}$
14. $\{bring(gm, gm), do(spg), before(so, sp)\}$
15. $\{bring(g, h), do(ml), do(st)\}$
16. $\{do(ml), do(st), do(spg)\}$
17. $\{do(fs), bring(g, h), bring(g, gm), do(ml)\}$
18. $\{bring(k_2, h), do(sp), bring(g, gm), do(ml)\}$
19. $\{do(fs), bring(k_2, h), bring(g, gm), do(ml)\}$
20. $\{do(fs), do(sp), bring(g, h), bring(g, gm)\}$
21. $\{bring(k_2, h), do(sp), bring(g, h), do(ml)\}$
22. $\{do(fs), do(sp), bring(g, gm), do(ml)\}$
23. $\{do(sp), bring(g, h), bring(g, gm), do(ml)\}$
24. $\{bring(k_2, h), do(sp), do(ml), do(so), do(spg)\}$
25. $\{do(fs), do(sp), bring(g, gm), do(so), not\text{-}t(gm, f)\}$
26. $\{do(fs), bring(k_2, h), do(ml), do(so), do(spg)\}$
27. $\{do(fs), bring(k_2, h), bring(g, h), do(ml), do(spg)\}$
28. $\{bring(gm, gm), bring(k_2, h), do(sp), do(ml), do(spg)\}$
29. $\{do(fs), bring(k_2, h), do(sp), do(ml), do(spg)\}$
30. $\{bring(gm, gm), do(fs), bring(k_2, h), do(ml), do(spg)\}$
31. $\{bring(gm, gm), do(fs), bring(g, h), do(ml), do(spg)\}$
32. $\{bring(gm, gm), do(fs), do(sp), bring(g, h), do(spg)\}$
33. $\{bring(gm, gm), do(fs), do(sp), do(ml), do(spg)\}$
34. $\{bring(gm, gm), do(sp), bring(g, h), do(ml), do(spg)\}$
35. $\{bring(gm, gm), do(fs), do(sp), do(so), do(spg), not\text{-}t(gm, f)\}$

ROVERS

Goals:

1. Uploaded rock image. *up*(r, i)
2. Uploaded rock sample. *up*(r, s)
3. Uploaded crater 1 image. (HG)
4. Uploaded crater 1 x-ray image. *up*(c_1, x)
5. Uploaded crater 1 sample. *up*(c_1, s)
6. Uploaded crater 2 image. *up*(c_2, i)
7. Uploaded crater 2 x-ray image. *up*(c_2, x)
8. Uploaded crater 2 sample. *up*(c_2, s)
9. Uploaded plain image *up*(p, i)
10. Uploaded plain sample. *up*(p, s)
11. Crater 2 image is uploaded before rock sample. *before*(c_2, i, r, s)
12. Plain sample is uploaded before rock image. *before*(p, s, r, i)
13. Crater 2 sample is uploaded before plain image. *before*(c_2, s, p, i)
14. Crater 2 x-ray image is uploaded before crater 1 sample. *before*(c_2, x, c_1, s)

MUGS:

1. $\{up(c_1, s), before(c_2, i, r, s)\}$
2. $\{up(c_1, s), before(c_2, s, p, i)\}$
3. $\{up(r, i), before(c_2, s, p, i)\}$
4. $\{up(c_2, i), before(p, s, r, i)\}$
5. $\{up(p, s), before(c_2, i, r, s)\}$
6. $\{before(p, s, r, i), before(c_2, x, c_1, s)\}$
7. $\{up(p, i), before(c_2, x, c_1, s)\}$
8. $\{up(c_2, s), before(p, s, r, i)\}$
9. $\{up(c_2, x), before(p, s, r, i)\}$
10. $\{before(p, s, r, i), before(c_2, s, p, i)\}$
11. $\{before(c_2, s, p, i), before(c_2, x, c_1, s)\}$
12. $\{up(c_2, x), before(c_2, s, p, i)\}$
13. $\{up(r, s), before(c_2, s, p, i)\}$
14. $\{up(c_1, x), before(c_2, s, p, i)\}$
15. $\{before(c_2, i, r, s), before(p, s, r, i)\}$
16. $\{up(p, i), before(c_2, i, r, s)\}$
17. $\{up(c_1, s), before(p, s, r, i)\}$
18. $\{up(r, i), before(c_2, x, c_1, s)\}$
19. $\{up(p, s), before(c_2, s, p, i)\}$
20. $\{before(c_2, i, r, s), before(c_2, s, p, i)\}$
21. $\{before(c_2, i, r, s), before(c_2, x, c_1, s)\}$
22. $\{up(p, s), before(c_2, x, c_1, s)\}$
23. $\{up(c_2, s), up(c_2, i), before(c_2, x, c_1, s)\}$
24. $\{up(p, i), up(c_2, x), up(c_2, s)\}$
25. $\{up(p, s), up(c_2, s), up(c_2, i)\}$
26. $\{up(p, i), up(c_2, x), up(c_1, x)\}$
27. $\{up(r, s), up(p, i), up(c_2, x)\}$
28. $\{up(p, s), up(c_2, x), up(c_2, i)\}$
29. $\{up(c_2, s), up(c_1, x), before(c_2, i, r, s)\}$
30. $\{up(p, s), up(c_2, s), up(c_1, x)\}$
31. $\{up(r, s), up(p, s), up(c_2, s)\}$
32. $\{up(p, i), up(c_2, s), up(c_1, x)\}$
33. $\{up(r, s), up(p, i), up(c_2, s)\}$
34. $\{up(p, s), up(p, i), up(c_2, s)\}$
35. $\{up(r, s), up(c_2, i), up(c_1, s)\}$
36. $\{up(r, i), up(p, s), up(c_1, s)\}$
37. $\{up(p, i), up(c_2, x), up(c_1, s)\}$
38. $\{up(r, i), up(p, i), up(c_2, x)\}$
39. $\{up(r, i), up(c_2, i), up(c_1, s)\}$
40. $\{up(r, s), up(c_2, s), up(c_1, s)\}$
41. $\{up(p, s), up(c_2, i), up(c_1, x)\}$
42. $\{up(r, s), up(p, s), up(c_2, i)\}$
43. $\{up(p, s), up(c_2, i), up(c_1, s)\}$
44. $\{up(r, i), up(p, s), up(c_2, i)\}$
45. $\{up(r, s), up(r, i), up(p, i)\}$
46. $\{up(r, s), up(p, i), up(c_1, s)\}$
47. $\{up(r, s), up(p, s), up(c_1, s)\}$
48. $\{up(r, i), up(c_1, x), before(c_2, i, r, s)\}$
49. $\{up(c_2, x), up(c_2, s), before(c_2, i, r, s)\}$
50. $\{up(p, s), up(c_2, x), up(c_2, s)\}$
51. $\{up(c_2, s), up(c_1, x), before(c_2, x, c_1, s)\}$
52. $\{up(r, s), up(c_2, s), before(c_2, x, c_1, s)\}$
53. $\{up(p, i), up(c_2, i), up(c_1, x)\}$
54. $\{up(r, s), up(p, i), up(c_2, i)\}$
55. $\{up(p, i), up(c_1, x), before(p, s, r, i)\}$
56. $\{up(r, s), up(p, i), before(p, s, r, i)\}$
57. $\{up(r, i), up(p, i), up(c_1, s)\}$
58. $\{up(c_2, x), up(c_1, x), before(c_2, i, r, s)\}$
59. $\{up(r, s), up(p, s), up(c_2, x)\}$
60. $\{up(p, s), up(c_2, x), up(c_1, x)\}$
61. $\{up(c_2, i), up(c_1, x), before(c_2, x, c_1, s)\}$
62. $\{up(r, s), up(c_2, i), before(c_2, x, c_1, s)\}$
63. $\{up(r, s), up(c_1, x), before(p, s, r, i)\}$
64. $\{up(r, s), up(c_1, x), before(c_2, x, c_1, s)\}$
65. $\{up(r, i), up(c_2, x), up(c_1, s)\}$
66. $\{up(p, i), up(c_2, s), up(c_1, s)\}$
67. $\{up(r, i), up(p, i), up(c_2, s)\}$
68. $\{up(p, i), up(c_2, i), up(c_1, s)\}$
69. $\{up(r, i), up(p, i), up(c_2, i)\}$
70. $\{up(p, s), up(c_2, s), up(c_1, s)\}$
71. $\{up(r, i), up(p, s), up(c_2, s)\}$
72. $\{up(r, i), up(c_2, s), before(c_2, i, r, s)\}$
73. $\{up(r, i), up(c_2, s), up(c_1, s)\}$
74. $\{up(p, s), up(p, i), up(c_2, x)\}$
75. $\{up(p, s), up(p, i), up(c_2, i)\}$
76. $\{up(r, i), up(c_2, x), before(c_2, i, r, s)\}$
77. $\{up(r, i), up(p, s), up(c_2, x)\}$
78. $\{up(p, s), up(c_2, x), up(c_1, s)\}$
79. $\{up(c_2, x), up(c_2, s), up(c_2, i), up(c_1, s)\}$
80. $\{up(r, i), up(c_2, x), up(c_2, s), up(c_2, i)\}$
81. $\{up(p, s), up(p, i), up(c_1, x), up(c_1, s)\}$
82. $\{up(r, i), up(p, s), up(p, i), up(c_1, x)\}$
83. $\{up(r, s), up(c_2, s), up(c_2, i), up(c_1, x)\}$
84. $\{up(r, s), up(c_2, x), up(c_2, i), up(c_1, x)\}$
85. $\{up(r, s), up(c_2, x), up(c_2, s), up(c_1, x)\}$
86. $\{up(c_2, x), up(c_2, s), up(c_2, i), up(c_1, x)\}$
87. $\{up(r, s), up(c_2, x), up(c_2, s), up(c_2, i)\}$
88. $\{up(r, s), up(p, s), up(p, i), up(c_1, x)\}$
89. $\{up(c_2, s), up(c_2, i), up(c_1, x), up(c_1, s)\}$
90. $\{up(r, i), up(c_2, s), up(c_2, i), up(c_1, x)\}$
91. $\{up(r, s), up(r, i), up(c_2, s), up(c_2, i)\}$
92. $\{up(c_2, x), up(c_2, i), up(c_1, x), up(c_1, s)\}$
93. $\{up(r, s), up(r, i), up(c_2, x), up(c_2, i)\}$
94. $\{up(c_2, x), up(c_2, s), up(c_1, x), up(c_1, s)\}$
95. $\{up(r, i), up(c_2, x), up(c_2, s), up(c_1, x)\}$
96. $\{up(r, s), up(r, i), up(c_2, x), up(c_2, s)\}$
97. $\{up(r, s), up(r, i), up(c_1, x), up(c_1, s)\}$
98. $\{up(r, s), up(r, i), up(p, s), up(c_1, x)\}$
99. $\{up(r, s), up(r, i), up(c_2, i), up(c_1, x)\}$
100. $\{up(r, s), up(c_2, x), up(c_1, x), up(c_1, s)\}$
101. $\{up(r, s), up(r, i), up(c_2, x), up(c_1, x)\}$
102. $\{up(r, s), up(r, i), up(c_2, s), up(c_1, x)\}$

APPENDIX B

APPENDIX CHAPTER 3

B.1 TASK RELAXATION INPUT DEFINITION

The Syntax to define relaxed tasks is defined below, along with an example of the definition of relaxed tasks and relaxed soft goals that are used as input for our extension of Fast Downward[1] Helmert [2006].

Relaxed Tasks The resource and time window constraint relaxations are provided to the planner as an additional input. For GSBNBp, the updates to the initial states for each relaxation must be specified as seen in the top of Figure 56. For ISSE, we must check each path for applicability in the current relaxation. For the relaxations considered here, it is sufficient to check only the last action. In a resource-relaxed task, an action is applicable if it does not consume more than is available in the relaxation. For example, if the most relaxed task has 16 units, then an action is only applicable in the relaxation with 4 units if it does not reduce the resource below 12. In TPP, there are two actions that consume resources `drive` and `buy`. The parameter constraints for each action and each relaxed task are defined as depicted in the bottom of Figure 56. All actions resulting in a remaining resource level (parameter 5) between 16 and 12 are applicable. The parameter definition of a constraint action allows wildcards $*$, to for example consider any `drive` action of truck `truck0` between any two locations and any initial fuel level. If an action is not constrained it is assumed to be applicable in any relaxation. In the case of time window relaxations, an action is applicable if the time lies within the time window of the relaxation. This is defined analogously to the constraint on the resource level, by constraining the parameters of actions within the time window.

Relaxation Soft Goals The relaxation soft goals described in Section 5.4.1 are defined as depicted in Figure 57. Due to the direct accessibility if the execution time point via the parameters of the actions in the grounded task, the formula for the time relaxation soft goals can be simplified to $\Box\neg v_{\text{not-app}}$, where $v_{\text{not-app}}$ indicates whether an action outside the time window has been applied.

[1] `https://github.com/aibasel/downward`

```
{
    "id": "4",
    "name": "money_truck0_level4",
    "inits": [
        "money(truck0, level4)"
    ],
    "upper_cover": [5],
    "lower_cover": [3]
}


{
    "id": "4",
    "name": "money_truck0_level4",
    "applicable": [
    {
        "name": "drive",
        "params": ["truck0", "*", "*", "*", "*"],
        "param_id": 5,
        "lower_bound": 12,
        "upper_bound": 16
    },
    {
        "name": "buy",
        "params": ["truck0", "*", "*", "*", "*"],
        "param_id": 5,
        "lower_bound": 12,
        "upper_bound": 16
    }],
    "upper_cover": [5],
    "lower_cover": [3]
}
```

Figure 56: Example for relaxed task definition for TPP, top resource relaxation and bottom time window relaxation. `inits` reflects the updates to the initial state and `applicable` specifies in which time/resource bound an action is applicable.

```
{
    "name": "relax_fuel_t0_level2",
    "type": "LTL",
    "formula": "G || fuel(t0,level15) || fuel(t0,level14) fuel(t0,level16)",
    "actionSets":[],
    "weaker": ["relax_fuel_t0_level3"],
    "stronger": ["relax_fuel_t0_level1"]
}

{
    "name": "relax_relay3_17-19",
    "type": "LTL",
    "formula": "G ! relay3_17-19_actions",
    "actionSets":[
        {
            "name": "relay3_17-19_actions",
            "actions":[
                {"name": "transmit", "params": ["rover", "relay3", "
                    visual_image", "level", "level15", "level", "level", "
                    level", "level"]},
                {"name": "transmit", "params": ["rover", "relay3", "
                    visual_image", "level", "level16", "level", "level", "
                    level", "level"]},
                {"name": "transmit", "params": ["rover", "relay3", "
                    visual_image", "level", "level20", "level", "level", "
                    level", "level"]},
                {"name": "transmit", "params": ["rover", "relay3", "
                    x_ray_image", "level", "level15", "level", "level", "
                    level", "level"]},
                {"name": "transmit", "params": ["rover", "relay3", "
                    x_ray_image", "level", "level16", "level", "level", "
                    level", "level"]},
                {"name": "transmit", "params": ["rover", "relay3", "
                    x_ray_image", "level", "level20", "level", "level", "
                    level", "level"]},
                {"name": "transmit", "params": ["rover", "relay3", "
                    soil_sample", "level", "level15", "level", "level", "
                    level", "level"]},
                {"name": "transmit", "params": ["rover", "relay3", "
                    soil_sample", "level", "level16", "level", "level", "
                    level", "level"]},
                {"name": "transmit", "params": ["rover", "relay3", "
                    soil_sample", "level", "level20", "level", "level", "
                    level", "level"]}
            ]
        }
    ],
    "weaker": ["relax_relay3_16-19","relax_relay3_17-20"],
    "stronger": ["relax_relay3_18-19"]
},
```

Figure 57: Example for relaxation soft goals. Top: resource relaxation soft goal where at most 2 units of fuel are consumed; bottom: time window relaxation soft goal where the upload to a relay is only possible between open time 17 and closing time 19.

BIBLIOGRAPHY

Timo Speith. A review of taxonomies of explainable artificial intelligence (XAI) methods. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 2239–2250. Association for Computing Machinery, 2022.

Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. The emerging landscape of explainable ai planning and decision making. *arXiv preprint arXiv:2002.11697*, 2020.

David Gunning and David Aha. DARPA's explainable artificial intelligence (XAI) program. *AI Magazine*, 40(2):44–58, 2019.

Horizon Europe. Research and innovation funding programme until 2027, 2021. URL `https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-europe_en`.

David Smith. Planning as an iterative process. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12)*, pages 2180–2185, Toronto, ON, Canada, July 2012. AAAI Press.

Cédric Pralet, David Doose, Julien Anxionnat, and Jérémie Pouly. Building resource-dependent conditional plans for an earth monitoring satellite. 32:490–498, 2022.

Benjamin Smith, William Millar, Julia Dunphy, Yu-Wen Tung, Pandu Nayak, Ed Gamble, and Micah Clark. Validation and verification of the remote agent for spacecraft autonomy. In *1999 IEEE Aerospace Conference. Proceedings (Cat. No. 99TH8403)*, volume 1, pages 449–468. IEEE, 1999.

Florian Geißer, Guillaume Povéda, Felipe Trevizan, Manon Bondouy, Florent Teichteil-Königsbuch, and Sylvie Thiébaux. Optimal and heuristic approaches for constrained flight planning under weather uncertainty. 30:384–393, 2020.

Bart M de Vries, Gerben JC Zwezerijnen, George L Burchell, Floris HP van Velden, Catharina Willemien Menke-van der Houven van Oordt, and Ronald Boellaard. Explainable artificial intelligence (xai) in radiology and nuclear medicine: a literature review. *Frontiers in medicine*, 10:1180773, 2023.

Patrick Weber, K Valerie Carl, and Oliver Hinz. Applications of explainable artificial intelligence in finance–a systematic review of finance, information systems, and computer science literature. *Management Review Quarterly*, 74(2):867–907, 2024.

Alberto Pozanco, Kassiani Papasotiriou, Daniel Borrajo, and Manuela Veloso. Combining heuristic search and linear programming to compute realistic financial plans. 33(1): 527–531, 2023.

Rita Matulionyte and Ambreen Hanif. A call for more explainable ai in law enforcement. In *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 75–80. IEEE, 2021.

Saad Khan, Simon Parkinson, Monika Roopak, Rachel Armitage, and Andrew Barlow. Automated planning to prioritise digital forensics investigation cases containing indecent images of children. 33(1):500–508, 2023.

Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

Markus Langer, Daniel Oster, Timo Speith, Holger Hermanns, Lena Kästner, Eva Schmidt, Andreas Sesing, and Kevin Baum. What do we want from explainable artificial intelligence (xai)?–a stakeholder perspective on xai and a conceptual model guiding interdisciplinary xai research. *Artificial Intelligence*, 296:103473, 2021.

Tathagata Chakraborti, Anagha Kulkarni, Sarath Sreedharan, David E Smith, and Subbarao Kambhampati. Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In *Proceedings of the international conference on automated planning and scheduling*, volume 29, pages 86–96, 2019a.

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54: 1–35, 2021.

Robert Hoffman, Shane Mueller, Gary Klein, and Jordan Litman. Measuring trust in the xai context. *PsyArXiv*, 2021.

Benjamin Krarup, Senka Krivic, Daniele Magazzeni, Derek Long, Michael Cashmore, and David E Smith. Contrastive explanations of plans through model restrictions. *Journal of Artificial Intelligence Research*, 72:533–612, 2021.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

Tim Miller. Contrastive explanation: A structural-model approach. *The Knowledge Engineering Review*, 36:e14, 2021.

Jörg Hoffmann and Daniele Magazzeni. Explainable AI planning (XAIP): Overview and the case of contrastive explanation (extended abstract). In *Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 2024, 2019, Tutorial Lectures*, pages 277–282. Springer International Publishing, 2019.

Felipe Rech Meneguzzi and Ramon Fraga Pereira. A survey on goal recognition as planning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*. ijcai.org, 2021.

Turgay Caglar and Sarath Sreedharan. Help! providing proactive support in the presence of knowledge asymmetry. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2024.

Gregory LeMasurier, Alvika Gautam, Zhao Han, Jacob W Crandall, and Holly A Yanco. Reactive or proactive? how robots should explain failures. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 413–422, 2024.

Pranut Jain, Rosta Farzan, and Adam J Lee. Co-designing with users the explanations for a proactive auto-response messaging agent. *Proceedings of the ACM on Human-Computer Interaction*, 2023.

Hendrik Baier and Michael Kaisers. Explainable search. In *2020 IJCAI-PRICAI Workshop on Explainable Artificial Intelligence*, 2020.

Ziyan An, Hendrik Baier, Abhishek Dubey, Ayan Mukhopadhyay, and Meiyi Ma. Enabling MCTS explainability for sequential planning through computation tree logic. In *ECAI 2024*. 2024.

Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. *arXiv preprint arXiv:1701.08317*, 2017.

Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Foundations of explanations as model reconciliation. *Artificial Intelligence*, 301:103558, 2021.

Sarath Sreedharan, Subbarao Kambhampati, et al. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS'18)*, volume 28, pages 518–526. AAAI Press, 2018a.

Sarath Sreedharan, Alberto Olmo, Aditya Prasad Mishra, and Subbarao Kambhampati. Model-free model reconciliation. *arXiv preprint arXiv:1903.07198*, 2019a.

Sarath Sreedharan, Subbarao Kambhampati, et al. Balancing explicability and explanation in human-aware planning. In *2017 AAAI Fall Symposium Series*, 2017.

Sarath Sreedharan, Tathagata Chakraborti, Christian Muise, and Subbarao Kambhampati. Expectation-aware planning: A unifying framework for synthesizing and executing self-explaining plans for human-aware planning. In Vincent Conitzer and Fei Sha, editors,

Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20), volume 34, pages 2518–2526. AAAI Press, January 2020a.

Sarath Sreedharan, Tathagata Chakraborti, Christian Muise, and Subbarao Kambhampati. Planning with mental models–balancing explanations and explicability. *Artificial Intelligence*, 2024.

Akkamahadevi Hanni, Andrew Boateng, and Yu Zhang. Safe explicable planning. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS'24)*. AAAI Press, 2024.

Stylianos Loukas Vasileiou, Ashwin Kumar, William Yeoh, Tran Cao Son, and Francesca Toni. Dr-hai: Argumentation-based dialectical reconciliation in human-ai interactions. *arXiv preprint arXiv:2306.14694*, 2023.

Sachin Grover, Sailik Sengupta, Tathagata Chakraborti, Aditya Prasad Mishra, and Subbarao Kambhampati. Radar: automated task planning for proactive decision support. *Human–Computer Interaction*, 35(5-6):387–412, 2020.

Valmeekam Karthik, Sarath Sreedharan, Sailik Sengupta, and Subbarao Kambhampati. Radar-x: An interactive interface pairing contrastive explanations with revised plan suggestions. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*, volume 35, pages 16051–16053. AAAI Press, February 2021.

Stylianos L Vasileiou, William Yeoh, and Tran Cao Son. A preliminary logic-based approach for explanation generation. In *Proceedings of the 2nd Worshop on Explainable Planning at ICAPS*, pages 132–140, 2019.

Alan Lindsay. Towards transparent planning and legible plan representations–a rule based planning approach. In *Proceedings of Workshop on Explainable Logic-Based Knowledge Representation at KR*, 2021.

Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan explicability and predictability for robot task planning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1313–1320. IEEE, 2017.

Aleck M MacNally, Nir Lipovetzky, Miquel Ramirez, and Adrian R Pearce. Action selection for transparent planning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1327–1335, 2018.

Alan Lindsay, Bart Craenen, Sara Dalzel-Job, Robin L Hill, and Ronald Petrick. Investigating human response, behaviour, and preference in joint-task interaction. In *Proceedings of the 3d Worshop on Explainable Planning at ICAPS*, 2020.

Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making hybrid plans more clear to human users - a formal approach for generating sound explanations. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings*

of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12). AAAI Press, 2012.

Sarath Sreedharan, Christian Muise, and Subbarao Kambhampati. Why did you do that? generalizing causal link explanations to fully observable non-deterministic planning problems. In *Proceedings of the 32th International Conference on Automated Planning and Scheduling (ICAPS'22)*. AAAI Press, 2022.

Sarath Sreedharan, Christian Muise, and Subbarao Kambhampati. Generalizing action justification and causal links to policies. pages 417–426, 2023.

Shirin Sohrabi, Jorge Baier, and Sheila McIlraith. Preferred explanations: Theory and generation via planning. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*, volume 25, pages 261–267. AAAI Press, 2011.

Andrew Murray, Benjamin Krarup, and Michael Cashmore. Towards temporally uncertain explainable ai planning. In *International Conference on Distributed Computing and Internet Technology*, pages 45–59. Springer, 2022.

Amanda Coles and Benjamin Krarup. Explaining plan quality differences. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS'24)*. AAAI Press, 2024.

Alan Lindsay and Ronald PA Petrick. Explaining plan selection through an analysis of object transition sequences. In *36th Workshop of the UK Planning and Scheduling Special Interest Group 2021*, 2021a.

Joseph Kim, Christian Muise, Ankit Shah, Shubham Agarwal, and Julie Shah. Bayesian inference of linear temporal logic specifications for contrastive explanations. In Sarit Kraus, editor, *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, pages 5591–5598. ijcai.org, 2019.

Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, volume 20, pages 81–88. AAAI Press, 2010.

Sarath Sreedharan, Siddharth Srivastava, David Smith, and Subbarao Kambhampati. Why can't you do that HAL? explaining unsolvability of planning tasks. In Sarit Kraus, editor, *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, pages 1422–1430. ijcai.org, 2019b.

Alan Lindsay. Towards exploiting generic problem structures in explanations for automated planning. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 235–238, 2019.

Alan Lindsay. Using generic subproblems for understanding and answering queries in xaip. In *Proceedings of the Worshop on Knowledge Engineering for Planning and Scheduling at ICAPS*, 2020.

Roykrong Sukkerd, Reid Simmons, and David Garlan. Tradeoff-focused contrastive explanation for MDP planning. In *29th International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2020.

Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Explain it as simple as possible, but no simpler–explanation via model simplification for addressing inferential gap. *Artificial Intelligence*, 2025.

Sam Toyer, Sylvie Thiébaux, Felipe Trevizan, and Lexing Xie. Asnets: Deep learning for generalised planning. *Journal of Artificial Intelligence Research*, 68:1–68, 2020.

Simon Ståhlberg, Blai Bonet, and Hector Geffner. Learning generalized policies without supervision using gnns. *arXiv preprint arXiv:2205.06002*, 2022.

Renee Selvey, Alban Grastien, and Sylvie Thiébaux. Formal explanations of neural network policies for planning. In *Proceedings of the 22th International Joint Conference on Artificial Intelligence (IJCAI'23)*, page 5446. International Joint Conferences on Artificial Intelligence Organization, 2023.

Silvia Tulli, Marta Couto, Miguel Vasco, Elmira Yadollahi, Francisco Melo, and Ana Paiva. Explainable agency by revealing suboptimality in child-robot learning scenarios. In *International Conference on Social Robotics*, pages 23–35. Springer, 2020.

Pascal Bercher, Susanne Biundo, Thomas Geier, Thilo Hoernle, Florian Nothdurft, Felix Richter, and Bernd Schattenberg. Plan, repair, execute, explain–how planning helps to assemble your home theater. In *Proceedings of the 24rd International Conference on Automated Planning and Scheduling (ICAPS'14)*, volume 24, pages 386–394. AAAI Press, 2014.

Martim Brandao, Amanda Coles, and Daniele Magazzeni. Explaining path plan optimality: Fast explanation methods for navigation meshes using full and incremental inverse optimization. In *Proceedings of the 31th International Conference on Automated Planning and Scheduling (ICAPS'21)*, volume 31, pages 56–64. AAAI Press, 2021a.

Khalid Alsheeb and Martim Brandao. Towards explainable road navigation systems. In *International Conference on Intelligent Transportation Systems*, 2023.

Sarath Sreedharan, Tathagata Chakraborti, Christian Muise, Yasaman Khazaeni, and Subbarao Kambhampati. –d3wa+–a case study of xaip in a model acquisition task for dialogue planning. 30:488–497, 2020b.

Alan Lindsay and Ron Petrick. Supporting explanations within an instruction giving framework. In *Proceedings of the 4th Worshop on Explainable Planning at ICAPS*, 2021b.

Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In Carles Sierra, editor, *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, pages 4829–4836. AAAI Press/IJCAI, 2018b.

Stylianos Loukas Vasileiou and William Yeoh. Please: Generating personalized explanations in human-aware planning. In *ECAI 2023*, pages 2411–2418. IOS Press, 2023.

Ashwin Kumar, Stylianos Loukas Vasileiou, Melanie Bancilhon, Alvitta Ottley, and William Yeoh. Vizxp: A visualization framework for conveying explanations to users in model reconciliation problems. 32:701–709, 2022.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *arXiv preprint arXiv:2409.13373*, 2024.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: Llms cant plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024.

Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

Christer Bäckström and Bernhard Nebel. Complexity results for SAS^+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535, 2009.

David E. Smith. Choosing objectives in over-subscription planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, pages 393–401, 2004.

Vitaly Mirkis and Carmel Domshlak. Abstractions for oversubscription planning. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, volume 23, pages 153–161, Rome, Italy, 2013. AAAI Press.

Carmel Domshlak and Vitaly Mirkis. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *Journal of Artificial Intelligence Research*, 52: 97–169, 2015.

Meysam Aghighi and Peter Jonsson. Oversubscription planning: Complexity and compilability. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, Austin, Texas, USA, January 2014. AAAI Press.

Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence Journal*, 129:5–33, 2001.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

James E Doran and Donald Michie. Experiments with the graph traverser program. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 294(1437):235–259, 1966.

Malte Helmert, Patrik Haslum, Jörg Hoffmann, et al. Flexible abstraction heuristics for optimal sequential planning. 2007.

Joseph C. Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

Stefan Edelkamp. Planning with pattern databases. In A. Cesta and D. Borrajo, editors, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 13–24. Springer-Verlag, 2001.

Carmel Domshlak, Jörg Hoffmann, and Ashish Sabharwal. Friends or foes? on planning as satisfiability and abstract cnf encodings. *Journal of Artificial Intelligence Research*, 36: 415–469, 2009.

Jendrik Seipp and Malte Helmert. Counterexample-guided Cartesian abstraction refinement. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, pages 347–351, Rome, Italy, 2013. AAAI Press.

Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29. AAAI Press, January 2015.

Jendrik Seipp, Florian Pommerening, and Malte Helmert. New optimization functions for potential heuristics. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 25, pages 193–201. AAAI Press, 2015.

Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. ieee, 1977.

Jorge A. Baier and Sheila A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In Yolanda Gil and Raymond J. Mooney, editors, *Proceedings of the 21st National Conference of the American Association for Artificial Intelligence (AAAI'06)*, pages 788–795, Boston, Massachusetts, USA, July 2006a. AAAI Press.

Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. AAAI Press/IJCAI, 2013.

Jorge A Baier and Sheila A McIlraith. Planning with temporally extended goals using heuristic search. In Derek Long and Stephen Smith, editors, *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS'06)*, pages 342–345, Ambleside, UK, 2006b. Morgan Kaufmann.

Stefan Edelkamp. On the compilation of plan constraints and preferences. In Derek Long and Stephen Smith, editors, *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS'06)*, pages 374–377, Ambleside, UK, 2006. Morgan Kaufmann.

Mark H Liffiton and Karem A Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *Journal of Automated Reasoning*, 40(1):1, 2008.

Joao Marques-Silva. Minimal unsatisfiability: Models, algorithms and applications. In *2010 40th IEEE International Symposium on Multiple-Valued Logic*, pages 9–14. IEEE, 2010.

Emilio Gamba, Bart Bogaerts, and Tias Guns. Efficiently explaining csps with unsatisfiable subset optimization. *Journal of Artificial Intelligence Research*, 78:709–746, 2023.

Michael Katz, Emil Keyder, Dominik Winterer, and Florian Pommerening. Oversubscription planning as classical planning with multiple cost functions. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, pages 237–245. AAAI Press, 2019.

Jorge A. Baier, Fahiem Bacchus, and Sheila A. McIlraith. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, 173(5-6):593–618, 2009.

Alfonso Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

Edmund M Clarke, Thomas A Henzinger, Helmut Veith, Roderick Bloem, et al. Handbook of model checking. volume 10. Springer, 2018.

Giuseppe De Giacomo, Riccardo De Masellis, and Marco Montali. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, Austin, Texas, USA, January 2014. AAAI Press.

Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

James Bailey and Peter J Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *Practical Aspects of Declarative Languages: 7th International Symposium, PADL 2005, Long Beach, CA, USA, January 10-11, 2005. Proceedings 7*, pages 174–186. Springer, 2005.

R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

Claude Berge. Hypergraphs north holland mathematical library. 1989.

Mark H Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. Fast, flexible mus enumeration. *Constraints*, 21:223–250, 2016.

Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

Kenneth L McMillan and Kenneth L McMillan. *Symbolic model checking*. Springer, 1993.

Jerry R. Burch, Edmund M. Clarke, David E. Long, Kenneth L. McMillan, and David L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, 1994.

Stefan Edelkamp and Peter Kissmann. Optimal symbolic planning with action costs and preferences. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1690–1695, Pasadena, California, USA, July 2009. Morgan Kaufmann.

Álvaro Torralba, Vidal Alcázar, Peter Kissmann, and Stefan Edelkamp. Efficient symbolic search for cost-optimal planning. *Artificial Intelligence*, 242:52–79, 2017.

Jendrik Seipp, Thomas Keller, and Malte Helmert. A comparison of cost partitioning algorithms for optimal classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pages 259–268. AAAI Press, 2017.

Jendrik Seipp and Malte Helmert. Counterexample-guided Cartesian abstraction refinement for classical planning. *Journal of Artificial Intelligence Research*, 62:535–577, 2018.

Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

classical domains. International planning competition instances, 2018. URL `https://github.com/AI-Planning/classical-domains`.

api.planning.domains. api.planning.domains, 2018. URL `https://api.planning.domains/`.

David Speck and Michael Katz. Symbolic search for oversubscription planning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*, pages 11972–11980. AAAI Press, February 2021.

Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge university press, 2004.

Sylvie Thiebaux, Jörg Hoffmann, and Bernhard Nebel. In defense of PDDL axioms. *Artificial Intelligence*, 168(1–2):38–69, 2005.

B. Cenk Gazen and Craig Knoblock. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In S. Steel and R. Alami, editors, *Proceedings of the 4th European Conference on Planning (ECP'97)*, pages 221–233. Springer-Verlag, 1997.

Bernhard Nebel. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, 12:271–315, 2000.

Alberto Camacho. Ltlfkit, 2017. URL `https://bitbucket.org/acamacho/ltlfkit/src/master/`.

Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. From Spot 2.0 to Spot 2.10: What's new? In *Proceedings of the 34th International Conference on Computer Aided Verification (CAV'22)*, volume 13372 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 2022.

Hootan Nakhost, Jörg Hoffmann, and Martin Müller. Resource-constrained planning: A monte carlo random walk approach. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 181–189. AAAI Press, 2012.

Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

Stefan Palan and Christian Schitter. Prolific.ac – a subject pool for misc experiments. *Journal of Behavioral and Experimental Finance*, 17:22–27, 2018.

Tathagata Chakraborti, Sarath Sreedharan, Sachin Grover, and Subbarao Kambhampati. Plan explanations as model reconciliation. In *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI'19)*, pages 258–266, 2019b.

Tathagata Chakraborti and Subbarao Kambhampati. (when) can ai bots lie? In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 53–59, 2019.

Sarath Sreedharan, Alberto Olmo Hernandez, Aditya Prasad Mishra, and Subbarao Kambhampati. Model-free model reconciliation. In Sarit Kraus, editor, *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, pages 587–594. ijcai.org, 2019c.

Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, David E Smith, and Subbarao Kambhampati. A bayesian account of measures of interpretability in human-ai interaction. *arXiv preprint arXiv:2011.10920*, 2020c.

Devleena Das, Siddhartha Banerjee, and Sonia Chernova. Explainable ai for robot failures: Generating explanations that improve user assistance in fault recovery. *arXiv preprint arXiv:2101.01625*, 2021.

Devleena Das, Been Kim, and Sonia Chernova. Subgoal-based explanations for unreliable intelligent decision support systems. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pages 240–250, 2023.

Maayan Shvo, Toryn Q Klassen, and Sheila A McIlraith. Resolving articleonceptions about the plans of agents via theory of mind. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pages 719–729, 2022.

Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Tldr: Policy summarization for factored ssp problems using temporal abstractions. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, pages 272–280. AAAI Press, 2020d.

Martim Brandao, Gerard Canal, Senka Krivić, Paul Luff, and Amanda Coles. How experts explain motion planner output: a preliminary user-study to inform the design of explainable planners. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 299–306. IEEE, 2021b.

Martim Brandao, Masoumeh Mansouri, Areeb Mohammed, Paul Luff, and Amanda Coles. Explainability in multi-agent path/motion planning: User-study-driven taxonomy and requirements. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 172–180, 2022a.

Joao Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov. On computing minimal correction subsets. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

Ulrich Junker. Preferred explanations and relaxations for over-constrained problems. In *Proceedings of the 19st National Conference of the American Association for Artificial Intelligence (AAAI'04)*, 2004.

Barry O'Sullivan, Alexandre Papadopoulos, Boi Faltings, and Pearl Pu. Representative explanations for over-constrained problems. In *Proceedings of the 22st National Conference of the American Association for Artificial Intelligence (AAAI'07)*, volume 7, pages 323–328. AAAI Press, 2007.

Sharmi Dev Gupta, Begum Genc, and Barry O'Sullivan. Finding counterfactual explanations through constraint relaxations. *arXiv preprint arXiv:2204.03429*, 2022a.

Bart Bogaerts, Emilio Gamba, Jens Claes, and Tias Guns. Step-wise explanations of constraint satisfaction problems. In *ECAI 2020*, pages 640–647. IOS Press, 2020.

Tias Guns, Emilio Gamba, Maxime Mulamba, Ignace Bleukx, Senne Berden, and Milan Pesa. Sudoku assistant–an ai-powered app to help solve pen-and-paper sudokus. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI'23)*, volume 37, pages 16440–16442. AAAI Press, 2023.

Johan De Kleer and Brian C Williams. Diagnosing multiple faults. *Artificial intelligence*, 32 (1):97–130, 1987.

Joao Marques-Silva and Inês Lynce. On improving mus extraction algorithms. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 159–173. Springer, 2011.

Ulrich Junker. Quickxplain: Conflict detection for arbitrary constraint propagation algorithms. In *IJCAI'01 Workshop on Modelling and Solving problems with constraints*, volume 4, 2001.

Alessandro Previti and Joao Marques-Silva. Partial mus enumeration. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI'13)*, volume 27, pages 818–825. AAAI Press, 2013.

Joao Marques-Silva and Alessandro Previti. On computing preferred muses and mcses. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 58–74. Springer, 2014.

Martim Brandao, Amanda Coles, Andrew Coles, and Jörg Hoffmann. Merge and shrink abstractions for temporal planning. In *Proceedings of the 32th International Conference on Automated Planning and Scheduling (ICAPS'22)*. AAAI Press, 2022b.

Marcel Steinmetz, Sylvie Thiébaux, Daniel Höller, and Florent Teichteil-Königsbuch. Explaining the space of SSP policies via policy-property dependencies: Complexity, algorithms, and relation to multi-objective planning. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS'24)*. AAAI Press, 2024.

Ilankaikone Senthooran, Matthias Klapperstueck, Gleb Belov, Tobias Czauderna, Kevin Leo, Mark Wallace, Michael Wybrow, and Maria Garcia de la Banda. Human-centred feasibility restoration in practice. *Constraints*, 28(2):203–243, 2023.

Guilhem Fouilhé, Rebecca Eifler, Sylvie Thiébaux, and Nicholas Asher. Conversational goal-conflict explanations in planning via multi-agent llms. In *Workshop on Planning in the Era of LLMs (LM4Plan) at AAAI*, 2025.

Rebecca Eifler, Daniel Fiser, Aleena Siji, and Jörg Hoffmann. Iterative oversubscription planning with goal-conflict explanations: Scaling up through policy-guidance approximation. In *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, 2024.

Niklas Lauffer and Ufuk Topcu. Human-understandable explanations of infeasibility for resource-constrained scheduling problems. In *Proceedings of the 2nd Worshop on Explainable Planning at ICAPS*, 2019.

Ilankaikone Senthooran, Matthias Klapperstueck, Gleb Belov, Tobias Czauderna, Kevin Leo, Mark Wallace, Michael Wybrow, and Maria Garcia de la Banda. Human-centred feasibility restoration. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In S. Chien, R. Kambhampati, and C. Knoblock, editors, *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, pages 140–149, Breckenridge, CO, 2000. AAAI Press, Menlo Park.

Jana Koehler. Planning under resource constraints. In *ECAI*, volume 98. Citeseer, 1998.

Biplav Srivastava, Subbarao Kambhampati, and Minh B Do. Planning the project management way: Efficient planning by effective integration of causal and resource reasoning in realplan. volume 131, pages 73–134. Elsevier, 2001.

Minh Do and Subbarao Kambhampati. Sapa: A domain-independent heuristic metric temporal planner. In *Sixth European Conference on Planning*, 2014.

Patrik Haslum and Héctor Geffner. Heuristic planning with time and resources. In *Sixth European Conference on Planning*, 2001.

Amanda Coles, M Fox, and D Long. A hybrid lp-rpg heuristic for modelling numeric resource flows in planning. *Journal of Artificial Intelligence Research*, 46:343–412, 2013.

David E Smith and Daniel S Weld. Temporal planning with mutual exclusion reasoning. In *Proceedings of the 11st International Joint Conference on Artificial Intelligence (IJCAI'99)*, volume 99, pages 326–337, 1999.

William Cushing, Subbarao Kambhampati, Mausam, and Daniel S Weld. When is temporal planning really temporal? In *Proceedings of the 19st International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 1852–1859, 2007.

Sergio Jiménez, Anders Jonsson, and Héctor Palacios. Temporal planning with required concurrency using classical planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, volume 25, pages 129–137. AAAI Press, 2015.

Sharmi Dev Gupta, Begum Genc, and Barry O'Sullivan. Finding counterfactual explanations through constraint relaxations. *arXiv preprint arXiv:2204.03429*, 2022b.

Jagriti Agrawal, Amruta Yelamanchili, and Steve Chien. Using explainable scheduling for the mars 2020 rover mission. In *Proceedings of the 3d Worshop on Explainable Planning at ICAPS*, 2020.

Peng Yu, Brian Williams, Cheng Fang, Jing Cui, and Patrik Haslum. Resolving over-constrained temporal problems with uncertainty through conflict-directed relaxation. *Journal of Artificial Intelligence Research*, 60:425–490, 2017.

Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 81–88. AAAI Press, 2010.

Songtuan Lin, Alban Grastien, and Pascal Bercher. Towards automated modeling assistance: An efficient approach for repairing flawed planning domains. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI'23)*. AAAI Press, 2023.

Salomé Eriksson, Gabriele Röger, and Malte Helmert. Unsolvability certificates for classical planning. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*, pages 88–97. AAAI Press, 2017.

Salomé Eriksson, Gabriele Röger, and Malte Helmert. A proof system for unsolvable planning tasks. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS'18)*, volume 28. AAAI Press, 2018.

Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.

A Jorge, Sheila A McIlraith, et al. Planning with preferences. *AI Magazine*, 29(4):25–25, 2008.

Alan Lindsay, Bart Craenen, and Ronald PA Petrick. Within task preference elicitation in net benefit planning. In *ICAPS 2021 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2021.

Sheryl Mantik, Minyi Li, and Julie Porteous. A preference elicitation framework for automated planning. *Expert Systems with Applications*, 208:118014, 2022.

Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. A novel iterative approach to top-k planning. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS'18)*. AAAI Press, 2018.

David Speck, Robert Mattmüller, and Bernhard Nebel. Symbolic top-k planning. In Vincent Conitzer and Fei Sha, editors, *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pages 9967–9974. AAAI Press, January 2020.

Michael Katz and Shirin Sohrabi. Reshaping diverse planning. In Vincent Conitzer and Fei Sha, editors, *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pages 9892–9899. AAAI Press, January 2020a.

Daniel Neider and Ivan Gavran. Learning linear temporal properties. In *2018 Formal Methods in Computer Aided Design (FMCAD)*, pages 1–10. IEEE, 2018.

Alberto Camacho and Sheila A McIlraith. Learning interpretable models expressed in linear temporal logic. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*. AAAI Press, 2019.

Zohar Manna and Amir Pnueli. A hierarchy of temporal properties (invited paper, 1989). In *Proceedings of the ninth annual ACM symposium on Principles of distributed computing*, pages 377–410, 1990.

Matthew B Dwyer, George S Avrunin, and James C Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international conference on Software engineering*, pages 411–420, 1999.

Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger. Specification patterns for robotic missions. *IEEE Transactions on Software Engineering*, 2019.

Wil MP van Der Aalst, Maja Pesic, and Helen Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, 23: 99–113, 2009.

Valentin Seimetz. Learning ltl plan properties from example plans. In *Masters Thesis at Saarland University*, 2020.

Michael Katz and Shirin Sohrabi. Reshaping diverse planning. In Vincent Conitzer and Fei Sha, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9892–9899. AAAI Press, January 2020b.

Mark Roberts, Adele Howe, and Indrajit Ray. Evaluating diversity in classical planning. In *Proceedings of the 24rd International Conference on Automated Planning and Scheduling (ICAPS'14)*, pages 253–261. AAAI Press, 2014.

Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. 2006.

Tuan Anh Nguyen, Minh Do, Alfonso Emilio Gerevini, Ivan Serina, Biplav Srivastava, and Subbarao Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 190:1–31, 2012.

Jean-Raphaël Gaglione, Daniel Neider, Rajarshi Roy, Ufuk Topcu, and Zhe Xu. Learning linear temporal properties from noisy data: A maxsat-based approach. In *Proceedings of the 19th International Symposium on Automated Technology for Verification and Analysis*, pages 74–90. Springer, 2021.

Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial intelligence*, 116(1-2):123–191, 2000.

Jörg Hoffmann and Stefan Edelkamp. The deterministic part of ipc-4: An overview. *Journal of Artificial Intelligence Research*, 24:519–579, 2005.

Luigi Bonassi, Giuseppe De Giacomo, Marco Favorito, Francesco Fuggitti, Alfonso Emilio Gerevini, and Enrico Scala. Planning for temporally extended goals in pure-past linear temporal logic. In *Proceedings of the 33th International Conference on Automated Planning and Scheduling (ICAPS'23)*, pages 61–69. AAAI Press, 2023.

Richard Howey, Derek Long, and Maria Fox. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 294–301. IEEE, 2004.

Caroline Lemieux, Dennis Park, and Ivan Beschastnikh. General ltl specification mining (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 81–92. IEEE, 2015.

Brennan Cruse and Christian Muise. Discrete time series clustering and delineation: A tree-based approach to linear temporal logic discovery. In *Proceedings of the 5th Worshop on Explainable Planning at ICAPS*, 2022.

R Michael Young, Martha E Pollack, and Johanna D Moore. Decomposition and causality in partial-order planning. In *AIPS*, pages 188–194, 1994.

Tathagata Chakraborti, Jung koo Kang, Francesco Fuggitti, Michael Katz, and Shirin Sohrabi. Interactive plan selection using linear temporal logic, disjunctive action landmarks, and natural language instruction. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI'24)*. AAAI Press, 2024.