Resolving Over-subscribed Temporal Planning Problems through Fluent Human-Robot Collaboration

Peng Yu

Ph.D Thesis Proposal

Department of Aeronautics and Astronautics
Massachusetts Institute of Technology

Thesis Committee:

Professor Brian Williams (Chair)
Professor Randall Davis
Professor Leslie Kaelbling

August 31, 2014

Abstract

There has been a rapid growth in the deployment of robots in the past decade. From aircraft assembly to home cleaning, robots have been working for us in many domains of our lives. However, there is one important barrier that limits their performance: these autonomous systems become very brittle when they fail to achieve what they were asked to do. No matter it is a sophisticated manufacturing robot, or a simple navigation app, little support can be provided from them for resolving the failures. Such over-subscribed situations are either caused by the robot's incomplete knowledge about available actions for all objectives, or inconsistent goals given by the human. For example, a factory manager may be given an overly ambitious production goal, which cannot be accomplished given the limited time and equipment in the factory. Due to the scale and complexity of the planning tasks in real world problems, it is nearly impossible for humans to find resolutions independently. Unfortunately, the automatic planners used to control these robots cannot provide any useful assistance but signal failure in such situations.

This is the essential issue that I would like to address in my PhD thesis: to develop an interactive decision support system that works like an intelligent and trustworthy teammate that can support humans in these situations. The autonomous agent, called Uhura, collaborates with humans to resolve over-subscribed planning problems. The key idea is to reduce the problem to a scale that can be easily handled by the humans, with fewer decisions and constraints to consider. Uhura works like an expert in plan diagnosis, which (1) automatically detects the cause of failure (2) succinctly explains them to the users and (3) presents preferred resolutions for repairing the broken plans. Only information that is key to the resolution will be communicated. In addition, Uhura also asks for inputs from the human during the collaborations. Due to the limitation of the decision aid, it may not be aware of all available options and the users' preference over different resolutions. The collaboration will help Uhura gain additional knowledge and resolve problems beyond its domain of knowledge.

Uhura is an integrated system with three core modules: a temporal planner for plan generation, a tester for validating plan consistency, and a user interface for all communications. The planner and tester are supported by a conflict-directed relaxation algorithm that can diagnose over-constrained plans: first detect conflicts of inconsistent goals and incomplete planning domains, then generate resolutions to address the conflicts and incompleteness. The resolutions are alternative plans, each of which relaxed some of the goals and planning domains. After diagnosis, Uhura will initialize a dialog with the user to explain the cause of failure and present resolutions. The user can either take Uhuras advice, or ask Uhura to find different alternatives with a reason. This failure-driven dialog is initialized each time one or more conflicts are detected that threaten the task plan, during both planning and execution stages.

In my thesis, I will be focusing on the application of Uhura in the following scenarios, which are in the domains of automated manufacturing and logistics planning:

- Personal Transportation: Uhura serves as a personal transit advisor to find and economic ways to commute in cities, and a management tool for public transit providers to reduce passenger flow during peak hours.
- Deep-sea Explorations: Uhura is incorporated as part of a mission advisor for oceanographers to manage high-risk expeditions. It assists the scientists in planning activities with large temporal uncertainty, and rescheduling tasks in unexpected failure situations using available resources.
- Factory-floor Manufacturing: Uhura is integrated in a production advisor for planning and scheduling activities on the factory floor. It provides quick assistance for repairing broken plans due to conflicting goals and equipments, and failures occurred in operations.

Contents

1 Introduction

Every day, as individuals we miss objectives and deadlines, because we try to do too much, and do not estimate the complexity and time of our tasks accurately. This can lead to anywhere from a major annoyance, such as not getting dinner ready on time, to a major catastrophe. The same situation also occurs frequently when we work with robots: we tend to ask them to do more than what they can actually achieve. Modern robotic systems become very brittle in these situations, since they are very bad at handling over-subscription and uncertainty. Unlike a real human being, these robots can hardly provide or communicate any support for resolving these problems, leaving their human teammates alone to address the issues. This has been a major barrier for human to trust their robotic systems and collaborate with them confidently.

This is the essential issue that my PhD thesis will address. The key idea is to detect and resolve the mismatch between what the user wants and what the robots can do. Scenarios like these can be framed as Over-subscribed Temporal Planning Problems, where no feasible plan exists that can meet all the requirements. They are usually caused by an overly ambitious set of objectives, or insufficient actions in the planning domain. It is not difficult to check the feasibility of a planning problem with modern planning algorithms: a planner will signal failure if no solution can be found. However, few planners can detect and return the cause of failure at the same time, and none is able to provide suggestions for repairing the planning problem. Because of the huge number of the possible alternatives and the complex process of checking their feasibility, it is impossible for the human to figure out good resolutions for these broken problems.

The objective of my thesis is to develop a decision aid that helps the human-robot team to resolve over-subscribed temporal planning problems. The decision aid, called Uhura, repairs the broken problem through a collaborative process that involves both the human and the robots. First, it automatically detects and explains the causes of the incompleteness/inconsistency, and computes alternative plans and objectives. The users only need to consider a much reduced problem over these alternatives, and handle fewer decisions with key insights provided by Uhura. This makes large planning problems more accessible for the human user to work with. Second, Uhura is aware of the deficiency of the robots' capability and knowledge of the robot. If necessary, it will ask the human to provide additional information or assitance for the robots to complete their tasks. Therefore, the collaboration can enable the team to tackle problems that cannot be solved by either the human or the robot independently.

For example, consider a manufacturing example where a factory manager is planning for a fleet of robots to assemble two wing boxes. Each assembly requires two procedures: aligning the skin to the wing box (Figure 1) and applying rivets to secure the pieces (Figure 2). These procedures are done by a team of three robots: two mobility platforms (X-WAM) and one riveting robot (Baxter). Due to an unexpected failure in the riveting robot, the assembly process has just been paused. Without the riveting robot, the current production plan becomes over-subscribed since the other two robots are not capable of riveting parts. The manager turns to the plan advisor, Uhura, for help on evaluating the impacts of the problem and recovery suggestions. Their conversation during the resolution process is presented below.

> **Uhura**: I cannot find a plan that satisfies all your requirements. Riveting is necessary for completing the wing box assembly, but none of the functional robots is capable of doing it.
>
> **Manager**: OK, what options do I have to resolve this problem?
>
> **Uhura**: Can you have the riveting robot repaired in an hour? We can still complete the tasks for today if it is available in the afternoon.

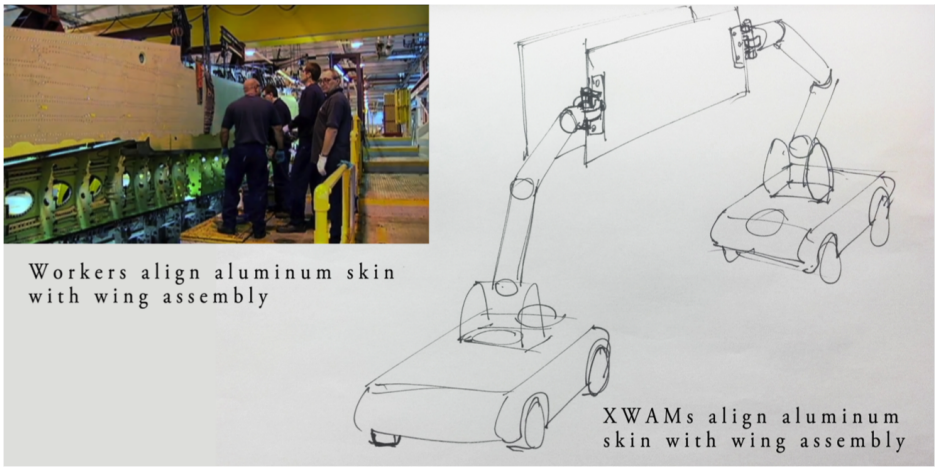

Figure 1: Aligning the Skin and the Wing Box

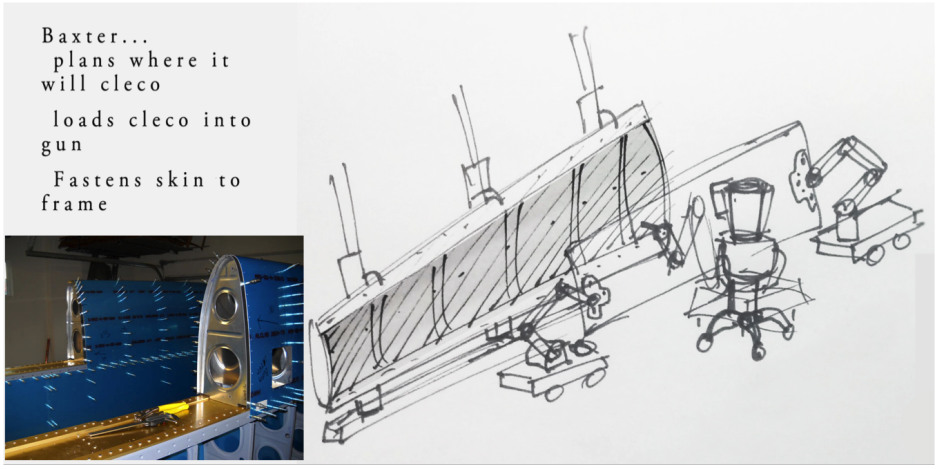

Figure 2: Riveting the Skin to the Wing Box Structure

Manager: No, the technicians are working on it but they said it would be offline for at least 12 hours.

Uhura: OK, can you call in one off-duty staff to help on the assembly today? If someone arrives in the afternoon we can complete the task by 8pm.

Manager: I can call in a staff to help, but the assemblies have to be ready before 7.

Uhura: Then could you join and help riveting the wing boxes before the staff arrives? You are qualified for working on the factory floor and we can then finish the assemblies on time.

Manager: OK, thanks Uhura.

This example demonstrates the collaborative resolution of a human-robot team on an over-subscribed planning problem, and highlights the desired features of Uhura during the process:

- Given an infeasible temporal planning problem, detect and explain the causes of failure, as either conflicting sets of planning goals, or incomplete domains of plan actions for achieving these goals.
- Compute and communicate preferred resolutions to the over-subscribed problem. A resolution can be a set of relaxations to the plan objectives, and/or a set of additional actions in the planning domain.
- Ask the users for assistance if the resolution cannot be found or accomplished by the robot alone. The assistance may be knowledge or preference information that is unknown to the robots, and/or physical actions that cannot be carried out by the robots.

Real world planning problems can be very difficult to solve due to their large number of constraints and the uncertainties in their models. These constraints are of different types for modeling requirements on state, time, resource and movements. To resolve over-subscription, Uhura has to perform at least three tasks: detect the causes of inconsistency and incompleteness, compute preferred resolutions that address all issues, and discuss them with the human to find. Therefore, Uhura is designed as a coordinated system of three modules with different functions, which work together to support the collaborative resolution between human and robots:

- A temporal planner for plan generation and validating problem completeness.
- A tester for validating plan consistency. If inconsistent, compute necessary relaxations that restore consistency.
- A user interface for communication with the users.

I have been working on the second module since the beginning of my master's work and developed an efficient tester. Given an inconsistent temporal plan, the tester is able to explain the cause of failure by extracting conflicts between constraints, and enumerate preferred relaxations for repairing the plan ([18]). In addition, it can handle uncertain temporal durations (set-bounded and probabilistic [16, 7]) and chance constraints in temporal plans, and make optimal trade-offs between risks taken and constraint relaxations. In real world scenarios, uncertainty in the models of planning problems and disturbances during plan execution is unavoidable. My approach has significantly improved the robustness of Uhura, and makes its solutions more robust for real world applications. This tester has been incorporated as part of a temporal plan executive, and has been used to support an autonomous air taxi driver system and a mission planner for deep-sea exploration missions of autonomous underwater vehicles.

In the next two years of my PhD research, I will complete the remaining two modules. First, a generative planner will be folded in for plan generation. This is the key to resolve over-subscribed planning problems caused by incompleteness. Given a specification of the planning domain, this planner is responsible for generating plans that meet all the users goals. If such a plan cannot be found, the planner will return a subset of the goals that cause the incompleteness, and a set of alternative and achievable goals that are similar to them. Uhura will then discuss these feasible alternatives with the user to find a preferred repair for the problem.

Second, a multi-modal user interface will be integrated for fluent collaboration between human and autonomous systems. This interface will support the following interactions during the resolution process:

- Understanding the goals and requirements of the planning problem from the users.
- Explain the cause of failure for an over-subscribed planning problem.

- Present relaxations and alternative goals to the over-subscribed problems to the users.
- Process user responses to the alternative proposals, including accepting, rejecting or rejecting with comments for modifications.

Effective communication is the key to address the mismatch between the humans requirements and the robots capabilities, thus the interface is required to be efficient and convenient to use. The interface will incorporate both natural language and a graphical touch display to achieve this goal. For example, it is easier to express goals and time constraints using verbal communications, while demonstrating changes to large-scale plans is much simpler through graphs. The objective is to make the interaction as natural as if it was with a human expert.

In this proposal, I will present my proposed approach for resolving over-subscrived planning problems, and discuss the remaining challenges and open questions in details. Section 2 presents the formal definition of the collaborative resolution for temporal planning problems. Section 3 presents my work on the tester and discusses my proposed approach for incorporating the temporal planner and user interface. Section 4 summarizes the current progress of my work and presents the schedule for accomplishing the remaining thesis objectives.

2 Problem Statement

Uhura is designed to work on temporal planning problems with constraints on time and states, and compute relaxations to the problem specifications to resolve any over-subscription. This is commonly referred as the class of temporal-metric-continuous problems in literature, and several formulations have been developed for it. For example, a commonly used one is simple temporal-metric planning problem, presented in [8]. It introduced many useful features, such as durative actions and time-independent metric change. Formally, a simple temporal-metric planning problem is defined as a 4-tuple $< I, A, G, M >$, where:

- I is the initial state: a set of propositions and an assignment of values to a set of numeric variables. Either of these sets may be empty.
- A is the planning domain that defines all allowed actions. Each action a is a 6-tuple $< dur, pre_{\vdash}, eff_{\vdash}, pre_{\leftrightarrow}, pre_{\dashv}, eff_{\dashv} >$, where:
  - $pre_{\vdash}$ and $pre_{\dashv}$ are the start and end conditions of a: they must hold at the time when a start or end.
  - $eff_{\vdash}$ and $eff_{\dashv}$ are the start and end effects of a: starting or ending a will update the world state according to these effects. A collection of effects $eff_x, x \in \vdash, \dashv$ consists of three items:
    - eff_x^-: propositions to be deleted from the world state;
    - eff_x^+: propositions to be added to the world state;
    - eff_x^n: effects acting on numerical variables;
  - $pre_{\leftrightarrow}$ are the invariant conditions of a that must hold at every point in the open interval between the start and end of a.
  - dur are the duration constraints of a, calculated on the basis of the world state in which a is started, and constraining the length of time that can pass between the start and end of a. They each refer ro the special parameter $?duration$, denoting the duration of a.

- G is the goal state: a set of propositions and conditions over numeric variables.
- M a metric objective function defined over the values of numeric variables at the end of the plan, and the special variable total-time, denoting the makespan of the plan.

This formulation provides the basis for modeling temporal planning problems, with constraints on linear resources. For Uhura, I will modify this model with two extensions: an augmented goal representation to allow goal states that evolve over time, and relaxable versions of goal states, planning domains and domain objects.

Time-evolved Goal Representation

The goal representation in the simple temporal metric planning problem is static: they must all hold at the end of a valid plan. However, in most real world problems, our objectives evolve over time, and we may have temporal restrictions between them. For example, in the manufacturing example, the technician would like all fixtures and wing boxes be unloaded within an hour, and no riveting is allowed until everything is in place. These features cannot be intuitively captured using a static goal representation.

In this thesis, I use Qualitative State Plans (QSPs,[10]) as the goal representation for the planning problem, which can model time-evolved goals given by the users. QSPs use episodes and temporal constraints to specify the users' objectives and requirements on states, resources and time. Formally, a QSP is represented as a 3-tuple $< E, EP, TC >$, where

- E is a set of events. Each event $e \in \mathcal{E}$ can be assigned a non-negative real value, and denotes a distinguished point in time.
- EP is a set of episodes. Each episode specifies an allowed state trajectory between a starting and an ending event. They are used to represent the state constraints. An episode, ep, is a tuple $< e_S, e_E, l, u, sc >$ where:
  - e_S and e_E in ep are the start and end events of the state trajectory.
  - l and u are the lower and upper bounds on the temporal duration of the episode.
  - sc is a state constraint that must hold true over the duration of the episode. In this thesis, the state constraint sc can be either a PDDL predicate, or a condition over numeric variables.
- TC is set of simple temporal constraints. It represents the temporal requirement between events in E, and can be viewed as a special type of episode without state constraint. A simple temporal constraint [6] is a tuple $< e_S, e_E, lb, ub >$ where:
  - e_S and e_E in E are the start and end events of the temporal constraint.
  - lb and ub represent the lower and upper bounds of the duration between events e_S and e_E, where $lb \leq$ Time(e_E) - Time(e_S) $\leq ub$ ($lb \in \mathbb{R} \cup -\infty$, $lb \in \mathbb{R} \cup +\infty$).

For example, the objectives and requirements in the manufacturing example can be represented by the following QSP (Figure 3). There are five state constraints: (in-place fixture), (in-fixture wingbox-1), (in-fixture wingbox-2), (assembled wingbox-1), (assembled wingbox-2). Both wing assemblies should be completed by the end of this plan. In addition, there are three temporal constraints in this QSP:

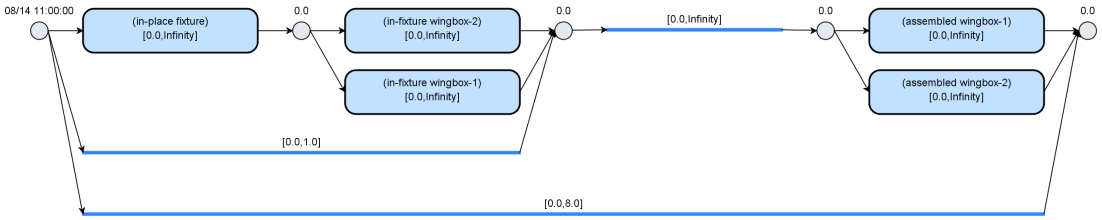

Figure 3: The Qualitative State Plan of the technician's production goals

- An overall temporal contraint, [0,8], that represents the production goal of completing all assemblies by 7pm.
- A constraint between (in-fixture wingbox-?) and (assembled wingbox-?) represents the technician's requirement of having all pieces ready before the final assembly.
- A constraint associated with both fixture ready and wingbox in-fixture, [0,1], that represents the requirement of having all pieces moved in within an hour.

Relaxable Planning Domain and Goals

As mentioned in the introduction section, if a temporal planning problem is over-subscribed, Uhura may resolve it by relaxing some of the goals and contraints in the problem. More specifically, for an over-subscribed problem, the relaxations for the constraints in the qualitative state plan and the action models in the planning domain allows a solution plan to be generated. To support these relaxations, an augmentation to the original problem definition is required for specifying (1) the goals and planning domains that are relaxable, (2) the domain of their relaxations, and (3) the cost associated with the relaxations.

Here I present the definitions of the relaxable versions of the qualitative state plans and planning domains that meet these criterias. First, a relaxable qualitative state plan (rQSP) is a QSP where some of the episodes in EP and temporal constraints in TC are relaxable. The definitions of events (E) remains unchanged. Relaxable episodes and temporal constraints are defined as the followings:

- A relaxable episode, ep_R, is a 10-tuple $< e_S, e_E, l, u, sc, RD_l, RD_u, RD_{sc}, f_{sc}, f_{tc} >$, where e_S, e_E, l, u, sc are the same as a regular episode. The additional elements, RD_l, RD_u, RD_{sc}, f_{sc} and f_{tc}, define the domain and cost associated with the relaxation to this episodes, where:
  - RD_l and RD_u are the domains of allowed relaxations for l and u: $RD_l \subseteq [-\infty, l]$ and $RD_u \subseteq [u, +\infty]$. They specify the range to what extent can l and u be relaxed: $Relax(l) \in RD_l$ and $Relax(u) \in RD_u$.
  - RD_{sc} is a set of state constraints, $sc_1, sc_2, ...$, which are alternatives to sc. It specifies the domain of allowed relaxations for the state constraint.
  - $f_{sc} : RD_{sc} \rightarrow R^+$ is a cost function that maps a relaxation to the state constraint, sc, to a positive cost value:
  - $f_{tc} : RD_l \cup RD_u \rightarrow R^+$ is a cost function that maps a relaxation to the lower or upper bounds, l or u, to a positive cost value.

- A relaxable temporal constraint, tc_R, is a 7-tuple $< e_S, e_E, lb, ub, RD_l, RD_u, f_{tc} >$, where e_S, e_E, lb, ub are the same as a regular temporal constraint. The additional elements, RD_l, RD_u and f_{tc}, define the domain and cost of relaxing the lower or upper bounds of this constraint. Their definitions are identical to the temporal relaxations of episode.

For example, in the manufacturing scenario presented in preceding section, the temporal constraint for plan completion can be made relaxable: the technician may be ok with a delay up to an hour. This constraint will have a relaxation domain of [8,9]. In addition, if the technician is flexible on the location of wingbox-1 during assembly, the state constraint of this episode, (in-fixture wingbox-1), could be relaxable with a domain [(in-cart wingbox-1), (on-ground wingbox-1)]. It means that the wingbox may be placed on the cart or on the ground prior to assembly, if it cannot be placed in the fixture. In the graphical representation of a QSP, relaxable temporal constraints are represented by dotted arrows, while relaxable episodes are represented by squares with dotted borders (Figure 4). All graphs in this proposal follow the convention.

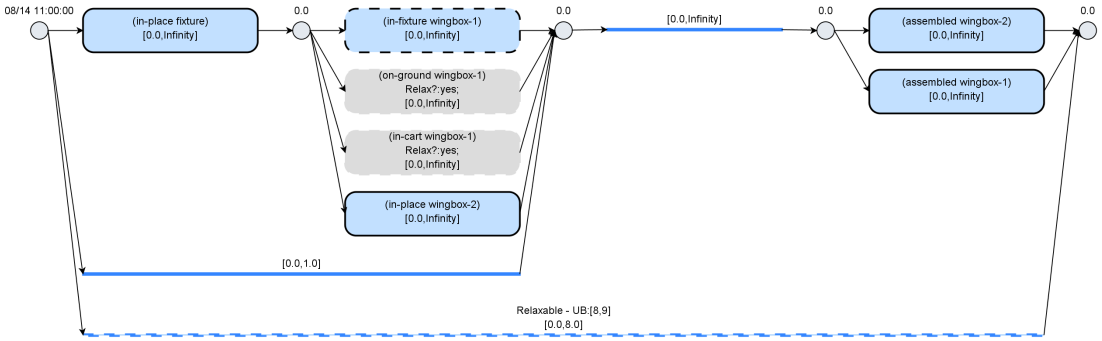

Figure 4: A relaxable QSP of the technician's production Goals

Second, a relaxable version of the planning domain defines allowed relaxations for the domain that can be used by Uhura to resolve over-subscription. A relaxation adds an additional action to the planning domain. Formally, a relaxable planning domain is defined as a 2-tuple $< A, A_{RX} >$, where:

- A is the set of initially allowed actions. This is the same as the planning domain in the original problem definition.
- A_{RX} is the domain of relaxations for A. It is a set of additional actions, which can be used in the plan if A is relaxed.

For example, the planning domain in the manufacturing scenario contains three actions: setup, unload and rivet (Figure 5). They are sufficient for the goal states defined in the original problem. A relaxable version of this planning domain defines two additional actions in its domain of relaxation, as shown in (Figure 6). They provide alternatives for the rivet action so that the technician and a backup staff could help complete the procedure. If relaxed, these actions can be added to the domain to support goals states that were previously unachieveable.

Finally, I define a relaxable version of the domain objects that allows additional objects to be added to the planning problem. This can be used to model relaxations over available resources and agents. Similar to relaxable planning domains, relaxable domain objects is defined as a 2-tuple $< O, O_{RX} >$, where:

```
(:durative-action setup
 :parameters (?f - fixture)
 :duration (= ?duration 0.5)
 :condition
     (and  (at start (available ?f)))
 :effect
     (and  (at start (not (available ?f)))
           (at end (in-place ?f))))


(:durative-action rivet
 :parameters (?r – rivet-robot
              ?w - wingbox)
 :duration (= ?duration 3.5)
 :condition
     (and  (at start (in-fixture ?w))
           (at start (available ?r)))
 :effect
     (and  (at start (not (available ?r)))
           (at end (available ?r))
           (at end (assembled ?w))))
```

```
(:durative-action unload
 :parameters (?f – fixture
              ?m1 – moving-platform
              ?m2 – moving-platform
              ?w - wingbox)
 :duration (= ?duration 0.25)
 :condition
     (and  (at start (on-cart ?w))
           (at start (in-place ?f))
           (at start (available ?1))
           (at start (available ?m2)))
 :effect
     (and  (at start (not (available ?m1)))
           (at start (not (available ?m2)))
           (at end (available ?m1))
           (at end (available ?m2))
           (at end (in-fixture ?w))))
```

Figure 5: Planning domain actions in the manufacturing scenario

```
(:durative-action rivetByBackupStaff
 :parameters (?s – backup-staff
              ?w - wingbox)
 :duration (= ?duration 3.5)
 :condition
     (and  (at start (in-fixture ?w))
           (at start (available ?s)))
 :effect
     (and  (at start (not (available ?s)))
           (at end (available ?s))
           (at end (assembled ?w))))
```

```
(:durative-action rivetByTechnician
 :parameters (?t – technician
              ?w - wingbox)
 :duration (= ?duration 3.5)
 :condition
     (and  (at start (in-fixture ?w))
           (at start (available ?t)))
 :effect
     (and  (at start (not (available ?t)))
           (at end (available ?t))
           (at end (assembled ?w))))
```

Figure 6: Actions in the domain of relaxation for same planning domain

- O is the set of initially allowed domain objects.
- O_{RX} is the domain of relaxations for O. It is a set of additional objects, which can be used in the plan if O is relaxed.

For example, in the manufacturing scenario, after the loss of the rivet robot, Uhura decided to relax the domain objects by suggesting the technician to ask an off-duty staff to complete the tasks. When the technician rejected this proposal, it further relaxed the domain by asking the technician to join the assembly task, in order to complete both wingboxes on time.

Note that the domains of relaxations for episodes, planning domains and domain objects are not static: they can be expanded during the planning process if necessary. The online expansion of relaxation domains provides much more flexibility for Uhura in resolving over-subscribed problems. For example, a family wants to have Chinese food for dinner and return home in 2 hours. However, there is no Chinese restaurants nearby that satisfies their temporal constraint.

In situations like this, Uhura may ask a recommendation system or a knowledge base, such as a search engine, for alternative dining locations that are not previously encoded in the problem. It may suggest the family to eat at a Thai restaurant, which is similar to Chinese restaurant and satisfies their timing requirement.

Solutions

A solution to a temporal planning problem is a set of grounded actions from the planning domain and timing constraints that specifies their start times, which transforms the initial state into a state satisfying the goal. A valid solution must respect all conditions, such as the precondition and duration of each action. In my thesis, I will use Temporal Plan Networks (TPNs) to encode solutions to temporal planning problems. This formalism was first introduced in [11] for temporal plans with flexibility and contingency. Formally, a TPN is defined as a 4-tuple, $< E, ACT, TC, e_{START} >$, where:

- E is a set of events. Each event $e \in E$ is assigned a non-negative real value, and denotes a specific instant in time. This concept is the same as event in QSPs.
- ACT is a set of activities between events, which represent grounded actions and their durations. Each activity, $act \in ACT$, is a 5-tuple $< e_S, e_E, l, u, act >$, where:
  - e_S and e_E are the start and end events of the activity.
  - l and u are the minimal and maximal duration of the activity.
  - act is a grounded action.

  More generally, an activity represents a state trajectory over its duration. The duration of the activity will be restricted by the temporal bounds, [l,u]. If $u > l$, this activity is a partial operator instantiation since the duration of this activity is flexible.
- TC is a set of simple temporal constraints. Like QSPs, simple temporal constraints in temporal plans specify the allowed durations between events. In addition, $\mathcal{TC}$ and $\mathcal{ACT}$ entail the temporal constraints between states in the QSP.
- $e_{START} \in E$ a distinct event that represents the first event in the temporal plan. e_{START} provides the timing reference for all other events in the plan.

For example, a temporal plan for the technician's planning problem is shown in (Figure 7). The temporal constraints from the QSP that specifies the technician's timing requirements are copied in this plan for reference. This plan contains five activities that setup the fixture, unload and rivet the wingboxes. It achieves all state constraints in the QSP (Figure 3). In addition, each activity is associated with a range of duration that is subsumed by the temporal constraints given in the planning domain. They satisfy all temporal constraints in the QSP.

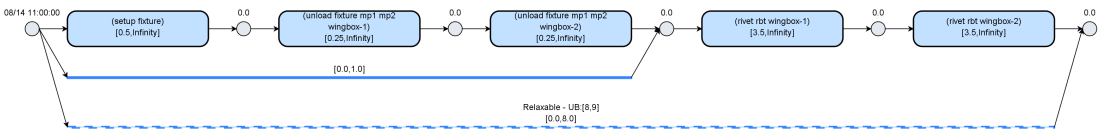

Figure 7: A plan for the manufacturing scenario

Over-subscription and Resolutions

Given a temporal planning problem, there might not be a solution that can transform the initial states into the goal states while respecting all the conditions. Such a problem is said to be **over-subscribed** since no valid plan can meet all the goals and requirements. The over-subscription of a temporal planning problem is the result of two reasons: **inconsistency** and **incompleteness**.

- Incompleteness is the result of a mismatch between actions in the planning domain and the desired goal states. In literature, such a planning problem is described as 'Relaxed Unsolvable', meaning that not all goals can be achieved even though the delete effects and temporal restrictions are removed from actions. For example, if we remove the object *rbt* from the planning domain (which simulates the loss of riveting robots in the manufacturing scenario), the problem becomes over-subscribed: we cannot perform the action 'rivet', which is the only way to achieve goal states (assembled wingbox-1) and (assembled wingbox-2).
- Inconsistency is caused by conflicting sets of goals and/or actions in a temporal plan. There are two types of inconsistencies due to conflicts in state and time.
  - State inconsistency refers to conflicts between the state constraints of actions and goals. For example, two actions are necessary to support some goal states, but their pre-conditions and effects may be mutually exclusive and prevent the plan from including both of them.
  - Temporal inconsistency refers to the conflicts between the temporal constraints in actions and goals. For example, the duration of activities between a pair of events may exceed the upper bound of a temporal constraint over the same events, eliminating all feasible schedules for the plan.

To resolve an over-subscribed temporal planning problem, one needs to relax the specification of the problem such that a valid plan can be found. For incompleteness, the resolution is to create new domain actions to bridge the gap between the initial conditions to the goal states, or to relax the goal states that cannot be achieved. The incompleteness is resolved if all state constraints in the relaxed QSP can be achieved by actions and objects in the relaxed domain. For inconsistency, the resolution also includes temporal relaxations for the constraints in the QSP, in addition to the options for resolving incompleteness. Inconsistency always comes with a conflicting set of goals, actions and constraints. The conflicts provide guidance for the resolution: we only need to examine and relax elements in the conflict for resolving inconsistency.

The resolution for an over-subscribed temporal planning problems can be summarized using a 3-tuple, $< RG, RA, RO >$, where:

- RG is a set of relaxations for the goal, which includes alternating or dropping temporal and state constraints in the QSP.
- RA is a set of relaxations for the planning domain, which are additional actions to be added to the domain.
- RO is a set of relaxations for the domain objects, which are additional objects that can be used in planning.

For example, in the manufacturing scenario, Uhura proposed two resolutions to address the technician's over-subscribed planning problem. The first one contains three relaxations to

the goal, the planning domain and the domain objects: (1) an additional action that allows a staff to rivet the wingbox; (2) an additional domain object that allows an off-duty staff to join the plan; and (3) a relaxation to the QSP that extends the time of completion requirement to 9 hours. These relaxation resolves the over-subscription and enables a feasible plan to be generated for the technician (Figure 8).

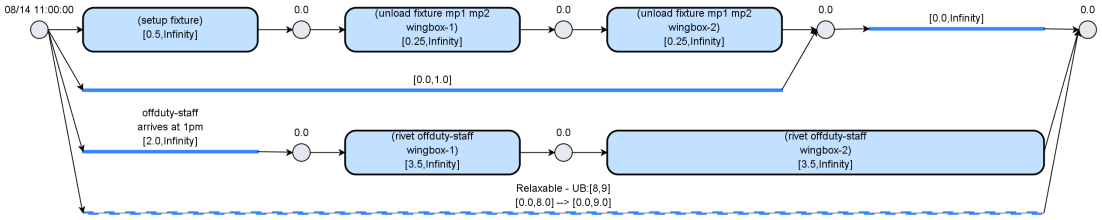

Figure 8: A plan for the manufacturing scenario

Uhura then proposed a second resolution based on the technician's response on not to delay the completion time. It further relaxed the planning domain and domain objects, by adding the technician and his riveting action to the problem. This allows the riveting process to start before the arrival of the off-duty staff, and enables a plan that completes two wingboxes without any delay (Figure 9).

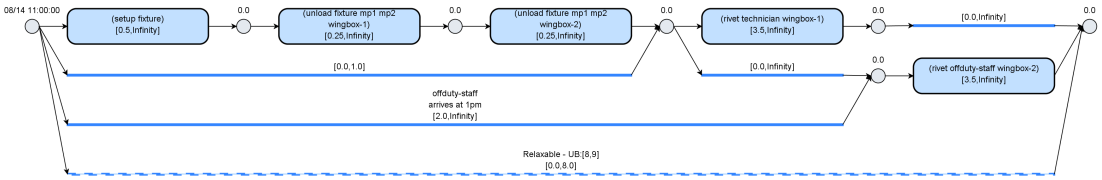

Figure 9: A plan for the manufacturing scenario

Given an over-subscribed problem, Uhura will work with the human collaboratively to find a preferred resolution, such that the human-robot team can achieve all their goals. The resolution may remove or modify goals, as well as elements in the domain specification. Note that the modifications to the action specification may apply to both parties in the team: Uhura may ask the user to complete some of the actions to meet all goals and constraints.

3 Approach

Section 1 demonstrates the expected behaviors of Uhura when working with humans to resolve over-subscribed planning problem. It makes robots intelligent, reliable and conversational team players that are trustworthy to their human teammates. From the aircraft assembly scenario, we can identify three features that are key for a fluent human-robot collaboration:

- Work collaboratively to resolve over-subscription, by assisting humans to detect cause of failures and proposing and assessing alternative options.
- Collaborate in a risk-sensitive manner. Propose courses of action that operate at acceptable risk levels. Pinpoint upcoming risks as the environment changes. Assist humans to make trade-offs between performance and risk-taken.

- Communicate easily through a conversational interface supported through multiple modalities. Include explanations and rationales for analyses, decisions and actions.

To support these festures, Uhura is designed as a coordinated system of three modules: planner, tester and user interface(Figure 10). The planner and tester reason over the temporal planning problem, and address the key research question of efficient resolution for over-subscription. The user interface handles all interactions with humans, and address the question of effective communication within human-robot teams on planning tasks. In this section, I will discuss my approach, current progress and remaining work on each module: first on the design and implementation of the planner and tester; then on the integration of the conversational user interface.

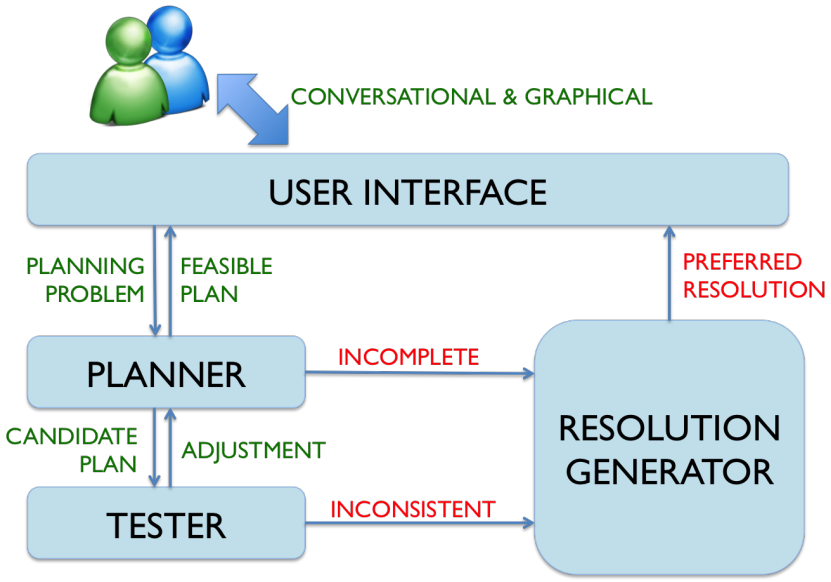

Figure 10: The architecture of Uhura

3.1 Resolving Over-subscribed Temporal Planning Problems

The planner and tester work together as the back end that generates plans and resolutions for the users. The input to them is a temporal planning problem that encodes the user's requirements and the model of the environment. The output is a feasible plan that satisfies all requirements, or a set of relaxations to some requirements and models if the problem is over-subscribed.

The approach I am taking is a two step procedure: first generating candidate plans that meet the user's goals with a planner, then evaluate the consistency of the candidate plans with a tester. If the input problem is incomplete, the planner will detect the cause of failure and work out a resolution with the resolution generator. If the candidate plan is inconsistent, the tester will find any conflicts that cause the inconsistency, either between temporal, resource, or precondition-effect constraints. Internally, the tester formulates an equivalent constraint satisfaction problem (CSP) for the plan that encodes the goals and constraints. It then uses a search algorithm to find consistent solutions to the constraint problem, and maps the solutions to the constraint satisfaction problem back to the planning problem. When applied for resolving

over-subscriptions, the equivalent CSP of a planning problem will be over-constrained, that is, no feasible set of variable assignments exist that can satisfy all constraints. The resolution to such a problem is a set of relaxed constraints and/or variable domains. When mapped back to the planning problems, the solutions becomes relaxations for the users' goals and planning domain specifications.

My work is divided into tree steps. The two steps focus on the tester: developing a relaxation algorithm for over-constrained problems with only temporal constraints, and then extending it to handle precondition-effect constraints, resource constraints and chance constraints. The third step focuses on the planner: it folds in the planner for generating plans and resolving any incompleteness in temporal planning problems.

- Step 1: Solve the over-constrained temporal problems. It is a simplified version of the tester problem in that it contains only temporal constraints, and its inconsistency is the result of conflicting sets of constraints. The resolution to such problems is a set of continuous relaxations for temporal constraints that resolves all conflicts. The simplicity of constraint types makes its domain of conflicts and relaxations much smaller. In addition, there might be uncertainty in some durations, and user specified chance constraints are allowed for specifying the bound on risk taken. Therefore, the relaxation may also include increasing chance constraint, which represents accepting a higher level of risk.

- Step 2: Solve the over-constrained vehicle routing problems with temporal constraints (VRPs). VRPs are widely used for modeling logistics, transportation and disaster relief problems. Compared to temporal problems, VRPs add types of constraints: resource and consistency constraints. Resource constraints model supply, demand and consumption of resources during traversals and visits. Consistency constraints over discrete variable assignments model restricted visits to locations and transit consistency. For example, the vehicle must be at the starting location before visiting any location, and return to the depot after its trip. The domain of relaxations is therefore extended to include increased resource, additional vehicles and new traversals between locations.

- Step 3: Fold in the temporal planner and solve over-subscribed temporal planning problems. The temporal planning problems can be viewed as a generalization to VRPs, with richer and more complex representations of goals, pre-conditions and effects. The domain of relaxations is extended to include both the planning domain specification and goals: a relaxation can add new actions to the planning domain, and/or modify some of the goals.

Testing the Consistency of Temporal Plans

As presented in the preceding section, the tester works on the candidate plans generated by the planner, and verifies their feasibility based on state, resource, temporal and risk consistency. The tester is a coordinated system that uses three different sub-testers to check all four types of consistency (Figure 11). First, the temporal consistency tester checks if the durations of activities satisfy all temporal constraints. If there are uncertain durations in the plan, the tester will also evaluate the risk and check if it is below the chance constraint. Second, the resource consistency tester checks if all resource constraints are satisfied. Finally, the state consistency tester checks if the pre-conditions and effects of all activities in the plan are consistent with each other. If a plan fails a test, the testers will return a conflict to the planner for a different candidate plan, and the resolution generator for a set of relaxations.

I began my work with the temporal tester for checking and restoring consistency of temporal problems. During my master's study, I developed a conflict-directed search algorithm that can enumerate preferred continuous relaxations in best-first order [18]. The algorithm,

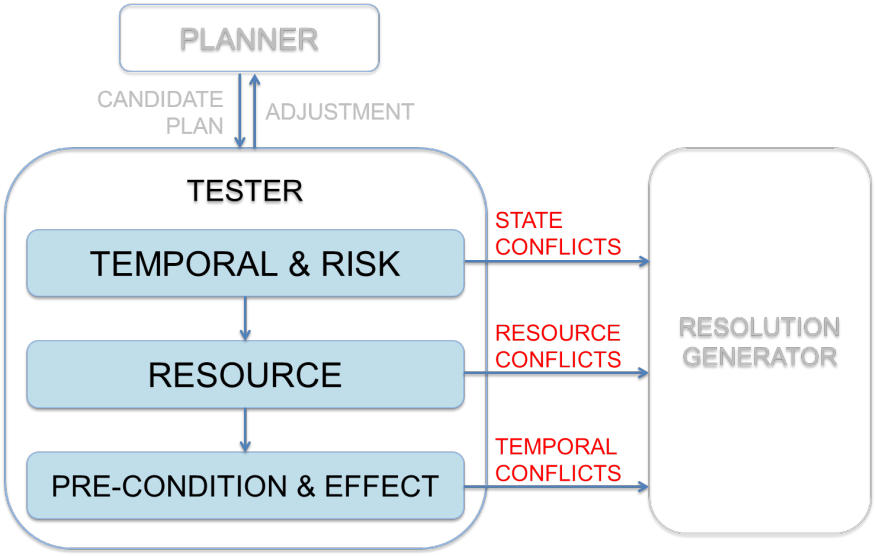

Figure 11: The architecture of the tester

called Best-first Conflict-Directed Relaxation (BCDR), learns conflicts from inconsistent temporal problems by detecting negative cycles. Each time a new conflict is learned, BCDR will compute a set of relaxations to some constraints that resolves all known conflicts. It repeats the learning-resolving procedures until (1) no more conflicts can be found, which indicates that the consistency has been restored; or (2) no resolution can be found to a learned conflict, which indicates tha the problem cannot be resolved. In addition, BCDR works with conditional temporal problems, whose constraints are guarded by discrete variable assignments. The conflict-directed framework is applicable to conflicts with both discrete assignments and continuous constraints, which enables BCDR to find optimal choices and relaxations at the same time.

Next, the approach was extended to handle temporal problems with uncontrollable duration. Such problems require the relaxation algorithm to restore controllability instead of consistency, which accounts for all possible outcomes of an uncontrollable duration. Their resolutions include not only relaxation for controllable duration, but also tightening for uncontrolable duration, which means taking more risk by restricting their outcomes. With a new conflict learning procedure that extracts minimal sets of conflicting constraints from uncontrollable problems (strong or dynamic), the same conflict-directed framework can be applied to uncertain durations [16]. The new algorithm, called Conflict-Directed Relaxation with Uncertainty (CDRU), enumerates preferred continuous relaxations for uncontrollable problems in best-first order. The resolutions generated by CDRU are more robust to disturbances during execution, making it applicable to many real world scenarios when uncertainty in temporal durations cannot be ignored.

Later, CDRU was further extended to handle chance-constrained temporal problems with probabilistic durations [17]. The controllability model assumes that all uncertain temporal duration are set-bounded with uniform distribution, while most real-world problems require different models of distributions over temporal uncertainty. In addition, the resolutions are often large sets of tightened duration and relaxed constraints, making it nearly impossible for users to understand the trade-offs being made between risks and requirements. The new problem formulation eliminates the assumptions on the temporal uncertainty and provides the users with intuitive and quantified values that indicate the risk being taken. Instead of

negotiating trade-offs between requirements and duration tightening, the extension presents the user with the value of risk taken explicitly, and specifies whether the chance-constraints need to be relaxed. The chance-constraint relaxation algorithm, called p-CDRU, uses the same conflict-directed resolution framework, and learns conflicts between temporal and chance constraints through a modified probabilistic scheduler [7]. With this new approach, Uhura can generate more suitable resolutions for different scenarios, and reduce the workload of users on real-world problems with large uncertainty.

The state consistency tester uses a causal link based approach to detect any inconsistency between preconditions and effects of activities. If a plan passes the check of the temporal tester, there must exist one or a set of execution strategies of activities that can meet all temporal constraints. The state consistency tester will try to find causal links between the initial conditions, goal states, preconditions and effects of activities within these execution strategies. This algorithm was first developed for Pike [12], a contingent plan executive that detects and repairs causal link violations. If inconsistency is detected, a set of violated causal links will be returned from the tester. They can be mapped back to a conflict between activities and temporal constraints, which is then used by the planner and resolution generator to adjust the candidate plans and compute relaxations.

Currently, I am working on the resource consistency tester. Algorithms for checking resource consistency have been incorporated in several temporal planners for problems with resource constraints, such as COLIN [2] and CRIKEY [3]. These approaches formulate the resource consistency problem as a LP and solve using a linear solver. My approach will extend these algorithms with the capability to detect conflicts in inconsistent plans, which are sets of resource constraints, temporal constraints and activities. These constraints can be resolved by the planner through a different candidate plan, or by the resolution generator through temporal and resource relaxations.

Generating Complete Candidate Plans

The next step in my PhD research is to fold in a generative planner and extend Uhura to resolve temporal planning problems. For a consistency problem, the plan is given and we only need the tester to check and restore the consistency between constraints. For a planning problem, Uhura has to generate a *complete* plan of actions that achieve all goals, and make sure that the plan is *consistent* with regards to user's requirements and domain models. From consistency to planning, the additional requirements of completeness add complexity to the relaxation problem: an infeasible planning problem may be caused by inconsistency and/or incompleteness, and the resolutions for them can be very different:

- Incompleteness is caused by insufficient actions in the planning domain. Such a planning problem is often described as 'Relaxed Unsolvable', meaning that not all goals can be achieved even though the delete effects are removed from actions. For incompleteness, the resolution is to create new domain actions to bridge the gap between the initial conditions to the goal states, or to alternate the goal states that cannot be achieved. Uhura needs to identify a set of unsupported goals in an over-subscribed planning problems, which is the indicator of incompleteness. An incomplete planning problem is resolved if all remaining goals can be achieved by actions in the domain specification.

- Inconsistency is caused by conflicting sets of constraints and activities in a plan. As presented in the preceding section, to resolve inconsistency, the tester continuously relaxes the constraints and/or activities involved in a conflict to eliminate it. The relaxations include not only temporal constraints and activity duration, but also resource, risk and

state constraints. The inconsistency is resolved once all conflicts are eliminated by the relaxations.

The addition of incompleteness makes the resolution of over-subscribed planning problems much more challenging. It requires significant changes to my previous approaches for developing the tester. Given an infeasible temporal planning problem, no planner can provide explanations on its cause of failure in terms of incompleteness and inconsistency. Prior work on repairing infeasible planning problems uses causal graphs to find excuses for the broken problems [9]. However, this is limited to classical planning problems and only the conflicts between initial and goals states, without the planning domains. Given an infeasible temporal planning problem, none of the modern temporal planners, like CRIKEY3 [3] and COLIN [2], is capable of returning any cause of failure or resolution.

Similar to cause of failure detection, resolution generation for over-subscribed planning problems is largely unexplored. There are algorithms for generating resolutions using the same causal graph approach [9], but the types of resolutions are restricted to adding initial states and dropping goal states. There is no approach that can continuously relax the temporal/risk/resource constraints in the goals. In addition, due to the large and undefined sets of possible changes to planning domain specifications, resolving infeasible problems using alternative actions is very difficult, and has been avoided by prior work on plan diagnosis and repair.

My approach is to fold a generative planner into Uhura. The planner is responsible for plan generation and incompleteness detection, and works with Uhura's tester to resolve over-subscribed planning problems. Even though there is no unified planning algorithm that has all desired capabilities for handling over-subscription, we do have modules and algorithms that can achieve each sub-task, including the followings:

- Generate complete plans and detect cause of incompleteness (Planner).
- Check and restore temporal and chance constraint consistency (Temporal & Risk Consistency Tester).
- Check and restore resource consistency (Resource Consistency Tester).
- Check and restore precondition-effect consistency (State Consistency Tester).

I am augmenting Uhura to be an integrated system of a planner and a tester suite, which are working in a coordinative manner to provide all capabilities for resolving over-subscribed temporal planning problems. The architecture of the planner and tester integration is presented in (Figure 10). The newly added planner is responsible for plan generation and detection of any incompleteness as unsupported set of goals. The tester suite, as presented in the preceding section, will work on the candidate plans generated by the planner. It evaluates the consistency of candidate plans in terms of time, risk, resource and precondition-effect. If any incompleteness is detected, the planner will pass the set of unsupported goals to the resolution generator, and ask the user for alternative goals or additional domain actions. If any inconsistency is detected by the testers, in addition to passing the conflict to the resolution generator for a relaxation, they also send the conflict back to the temporal planner and ask for an alternative candidate plan. The negation of the conflict will be encoded as a landmark so that the new plan generated by the planner is guaranteed to avoid this conflict.

Finding Good Resolutions

A good resolution to an over-subscribed planning problem minimizes perturbation to the user's requirements and the domain specifications. As demonstrated in the example of Section 1, the

key to good resolutions is not dropping goals or removing requirements, but finding alternatives for them. Remember from the problem statement in Section 2 that there are three types of resolutions:

- Relaxations for the goals, which include alternative temporal, risk, resource, and state constraints in the qualitative state plan.
- Relaxations for the planning domain, which are additional actions to be added to the domain.
- Relaxations for the domain objects, which are additional objects that can be used in planning.

All resolution types share one common feature: *similar alternatives*. For temporal relaxations, the domain is restricted to continuous relaxation or tightening of temporal bounds. The similarity is measured by the changes applied to the bounds, and the objective is to find minimal continuous relaxations that are necessary for restoring feasibility. The same principle applies to relaxations over resource and chance constraints. However, for relaxations of goal states and planning domain, there is no clear standard to measure the 'similarity' between alternatives. Therefore, I introduced a new type of relaxations for goals and actions, called **semantic relaxation**. The objective is to use semantic information to measure the *distance* between concepts, and find semantically similar alternative goals and actions to repair the over-subscribed problem. This is similar to the reasoning process used by human beings. For example, a student wants to have lunch at a Chinese restaurant, but the place is too far away and may cause him to miss the afternoon classes. There are only two restaurants near campus: one serves Thai food and the other one is Mexican. Given that Thai and Chinese restaurants are similar in that they both serve Asian style cuisine, a good suggestion for them would be to dine at the Thai restaurant, instead of the Mexican restaurant.

When applied to actions in the planning domain, semantic relaxations can suggest alternative actions with similar preconditions and effects for resolving incompleteness. For example, a family is preparing a Thanksgiving dinner. They would like to make a roast turkey but only have the recipe for chicken. Uhura will notice the semantic similarity between the concepts of chicken and turkey, and generate a relaxation that suggest the users to try applying the chicken recipe on turkey.

In summary, both incompleteness and inconsistency can be repaired by relaxations over the goals and planning domain specifications. For my PhD research, I defined the following two research questions that are key for generating good resolutions:

- Extract and continuously resolve conflicts from inconsistent (temporal, risk, resource and precondition-effect) candidate plans.
- Generate semantically similar alternatives for goal states and actions to resolve incompleteness and inconsistency.

I will be working on two questions in parallel. My approach for the first question will build on my master's work, with extensions to support resource constraints. The same conflict-directed framework for temporal relaxations remains applicable: the resource consistency tester will extract minimal conflicts over resource and temporal constraints. The resolution is formulated as a LP problem, whose solutions are the relaxations to the temporal bounds and/or resource requirements in the candidate plan. For precondition-effect conflicts, the resolution would be

to drop the threads of activities with broken causal links, or ask the planner for an alternative plan. In the latter case, the broken causal links are encoded as landmarks for the planner so that the same conflict will be avoided in all future candidate plans.

For the second question, my approach is to build a recommendation system that works with the human to find good alternatives for goals and actions. This is joint work with Jonathan Raiman. The recommendation system is supported by two algorithms: (1) a filtering algorithm that finds similar concepts in a candidate database [13]; and (2) a sequential diagnosis engine that iterates with the user to narrow down the range of candidates [5]. The filtering algorithm is like an autoencoder, which generates a series of filters from a database of alternative candidates. It takes the description of a concept as input, then finds candidates with similar descriptions in the database. For example, the description of a Chinese restaurant often consists of popular dishes in the cuisine. By finding concepts in the database or online with similar descriptions, a set of candidates can be retrived that are similar to it.

Given a set of similar alternatives, the sequential diagnosis engine will then initiates a dialog with the human to find the best alternative. It narrows down the range of candidates by asking a series of probing questions to the user to differentiate between them. This collaborative process will enable the human to contribute, by adding options that are unknown to the system, or by rejecting suggestions that are not preferred.

3.2 Interacting with Users Naturally

Effective communication is key for effective collaboration between team players. This applies to both human-human and human-robot teams. Communication creates transparency between teammates: it exposes and helps to resolve mismatches between their goals and constraints. The objective of Uhura is to generate plans that satisfy the users' need, and suggest resolutions for repairing over-subscribed planning problems. This cannot be done without a clear understanding of the user's requirements and preferences. In addition, the knowledge of Uhura is very limited compared to the human's knowledge and experience. Under some scenarios, the algorithm may run out of options, and the helps from humans would be key for resolving the problems. An effective dialogue component for a robot should support the following functions:

- Interpret the goals, requirements and preferences given by the human teammates.
- Communicate reasons for Uhura's decisions and actions.
- If an over-subscription is detected, provide explanations on the causes and their resolutions.
- If Uhura encounters a failure and runs out of option, ask the human for help.

Conversation is a natural mode of interaction for humans, and a key element of effective user interface. For example, I have worked on integrating a dialog manager [14] into a robotic air taxi system to support fluent driver-passenger interaction [15]. A conversational UI may also be supported by graphical touch displays for faster interaction and demonstration of abstract concepts, such as the sketching interface presented in [4]. There is no single mode of communication that is preferred in all scenarios. Graphs are intuitive and efficient for presenting alternative plans to the user, and an integrated click and drag interface would be very natural for the user to make minor tweaks to the plan or their requirements. On the other hand, verbal communication is more efficient for presenting comparative options, and asking the users for help. The interface should be multi-modal and support all verbal, visual and gesture communications.

An area of significant open research is conversational interfaces centered around defining, planning and executing tasks, through a deep connection between dialogue and collaborative methods for planning, execution, monitoring and diagnosis. The interface should not be just a translator of the planners outputs. Instead, it should be part of the planning process, representing the humans participation in all phases of this process. Using this interface, when the robots planner needs to collect more information or ask for assistance, the planner may initiate a conversation to address the issue.

An ideal device for implementing such an interface of Uhura would be a tablet: the display provides graphical interface; the touch screeen supports gesture inputs, such as clicking, dragging and scaling; and the microphone and speaker, plus the voice recognition and text-to-speech service supports the natural language communication. There are many other types of human-computer interactions, such as body gesture. To restrict the scope of my thesis, I will not be focusing on other options in my PhD research.

In summary, I will be working on the following two research questions on natural user interface design in support of Uhura's plan relaxation capability: (1) a graphical interface that renders charts for representing plans and resolutions, and recognizes the users' touch inputs; (2) a dialog management system that generates questions, parses the users' verbal inputs, and assembles responses. The interface will be implemented as a tablet app that integrates both capabilities.

4 Proposed Schedule

The current time is Summer 2014, and I am planning to defend my thesis by the end of Spring 2016. There will be three groups of milestones towards my thesis in the next two years. I will start with extending my current approach to support over-constrained vehicle routing problems, by adding the resource and precondition-effect consistency tester. Then I will move on to fold in the planner for resolving temporal planning problems. In addition, I will be working on the integration of dialog and tablet interaction for the user interface, and extend it to support interactions on negotiations of new types of resolutions. I will build it based on two prior works: the dialog manager for a Personal Transportation System [15] and the GUI for a flexible manufacturing system [1].

Each milestone is equivalent to a conference level publication: the conference deadlines provide constraints for monitoring my progress, and the feedbacks from reviewers will be used for evaluating my research. In the next six semesters, I will be completing one milestone per semester on average, and leave the Winter and Spring of 2016 for thesis writing. The schedule is summaried in the following list. Current work in progress are highlighted in bold with expected conferences and deadlines.

- **Summer 2014**: Resolving over-constrained probabilistic temporal problems through chance constraint relaxation (AAAI 2015).
- **Summer 2014**: Finding semantically similar alternatives for planning goals and actions (IJCAI 2015).
- **Fall 2014**: Resolving over-constrained vehicle routing problems through temporal and resource relaxation (ICAPS 2015).
- Spring 2015: Resolving over-subscribed temporal planning problems through relaxing goal and domain specifications.
- Winter 2016: Thesis draft completion.

- Spring 2016: Thesis approved and defended.

References

[1] Sean Burke, Enrique Fernandez, Luis Figueredo, Andreas Hofmann, Chris Hofmann, Erez Karpas, Steven Levine, Pedro Santana, Peng Yu, and Brian Williams. Intent recognition and temporal relaxation in human robot assembly. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*, 2014.

[2] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Researchcial Intelligence Research (JAIR)*, 44:1–96, 2012.

[3] Andrew Coles, Maria Fox, Derek Long, and A Smith. Planning with problems requiring temporal coordination. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08)*.

[4] Randall Davis. Magic paper: Sketch-understanding research. *Computer*, September 2007.

[5] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[6] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[7] Cheng Fang, Peng Yu, and Brian Williams. Chance-constrained probabilistic simple temporal problems. In *Proceedings the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, 2014.

[8] 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.

[9] Moritz Gbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10)*, 2010.

[10] Andreas G. Hofmann and Brian C. Williams. Robust execution of temporally flexible plans for bipedal walking devices. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS-2006)*, 2014.

[11] P. Kim, B. C. Williams, and M. Abramson. Executing reactive, model-based programs through graph-based temporal planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 487–493, 2001.

[12] Steve Levine and Brian Williams. Concurrent plan recognition and execution for human-robot teams. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*, 2014.

[13] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, 2011.

[14] Baoshi Yan, Fuliang Weng, Zhe Feng, Florin Ratiu, Madhuri Raya, Yao Meng, Sebastian Varges, Matthew Purver, Feng Lin, Annie Lien, Tobias Scheideck, Badri Raghunathan, Rohit Mishra, Brian Lathrop, Harry Bratt, and Stanley Peters. A conversational in-car dialog system. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2007.

[15] Peng Yu. Continuous relaxation to over-constrained temporal plans. Master's thesis, Massachusetts Institute of Technology, 2013.

[16] Peng Yu, Cheng Fang, and Brian Williams. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*, 2014.

[17] Peng Yu, Cheng Fang, and Brian Williams. Resolving over-constrained probabilistic temporal problems. In *submitted to the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[18] Peng Yu and Brian Williams. Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 2429–2436, 2013.