Programs with Flexible Time, Choice, and State

Choice?

Steve Levine
February 17th, 2016
Assignments

Problems Sets:

• Pset 1 due tonight at 11:59pm

• Pset 2 released tonight (Scheduling)

• Backup your work before updating!

Interesting references:

• ITC: I-hsiang Shu, Robert Effinger, Brian Williams, Enabling Fast Flexible Planning through Incremental Temporal Reasoning with Conflict Extraction.
Today: Combining what we’ve learned

Flexible Time

Execution monitoring

and

Flexible Time

Conflict-directed A*
Outline

• Flexible time + execution monitoring
  – Extracting causal links
  – Dispatching & monitoring

• Flexible time + conflict-directed A*
  – Plans with choice
  – Making optimal choices
Adding more flexibility

PLANS WITH CHOICE & TIME
Simple Temporal Network (STN)

Simple Temporal Network (STN):

Read: “Event B must occur between lowerbound and upperbound time after A”

Read: “Event B must occur between 5 and 10 [seconds] after A. 
AND Event C must occur between 2 and 3 [seconds] after C.”
Temporal Plan Network (TPN)

Temporal Plan Network (TPN):

Idea 1: Durative Activities

Read: “Start Activity at Event A and end it at Event B.”
Alt: “Activity will take between lb and ub time to execute.”

Idea 2: Decision Events: execute only one outgoing episode

Read: “After Event C, execute either Event A or Event B, depending on which episode has the lowest cost.”
Temporal Plan Network

Temporal Plan Network (TPN):

Idea 3: Hierarchical Composition

Decision/Choice

Sequential

Parallel

Start

End
Robust Program and Plan Execution

```
imageScienceTargets(Rover1, Rover2)
Parallel{
Sequence{
[5,10] Rover1.goto(p4);
[5,10] Rover1.goto(p5);
[2,5] Rover1.imageTargets1();
[5,10] Rover1.goto(p3);
}
Sequence{
[5,10] Rover2.goto(p1);
[5,10] Rover2.imageTargets2();
[2,5] Rover2.goto(p2);
[5,10] Rover2.goto(p3);
}
}
```

What if we want more flexibility in our plan?
A Different Choice
A Plan with Choice

Assume for this plan, that edges without explicit temporal constrains are [0,0].
A Plan with Choice

```plaintext
Parallel{
  Sequence{
    [5,10] Rover1.goto(p4);
    [5,10] Rover1.goto(p5);
    Choice{
      [2,5] Rover1.imageTargets1();
      [5,10] Rover1.goto(p3);
    }
  }
  Sequence{
    [5,10] Rover2.goto(p1);
    Choice{
      [2,5] Rover2.imageTargets2();
      [2,5] Rover2.goto(p2);
    }
  }
}
```

RMPL
What can Influence the Choice?
What can Influence the Choice?

A utility associated with the choice.
Temporal Consistency: Some choices result in temporally infeasible plans.

(In order for the rovers to rendezvous) One choice influences the other choice.
With choices we can represent…

• Alternative ordering of actions
• Alternative methods for competing a task
• Alternative resource assignments
• Alternative task assignments (who does what)
When can the Choice be made?

1. Describe Temporal Plan
2. Test Consistency
3. Schedule Plan
4. Execute Plan

Make Choices

Make Choices
A single “cognitive system” language and executive.

User

Goals & models in RMPL

Enterprise

Uhura

Collaboratively resolves goal failures

Kirk

Sketches mission and assigns tasks

Burton

Plans actions

Pike

Coordinates and monitors tasks

Sulu

Plans paths

Bones

Diagnoses likely failures

Control Commands

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/.
Example TPN
Choosing as a Constraint Satisfaction Problem
Assign variables A, B, & C such that all of the temporal constraints with true guards are temporally consistent.
Different candidate subplans of TPN

- Find which edges have activated guards
Different candidate subplans of TPN
Trace Trajectories
Check Schedulability

• Don’t test consistency at each step.

⇒ Only when a path induces a cycle, check for negative cycle in the STN distance graph
Check Schedulability

- Example: Inconsistent $\rightarrow$ Conflict!
Trace Alternative Trajectories

• Complete paths
Are all choices consistent?

- (Board)
Another example: Breakfast
How to get conflicts

• In problem set / polycell:
  – Conflicts came from DPLL + unit propagation

• Here:
  – Conflicts come from negative cycle detection (through ITC – Incremental Temporal Consistency)
ITC Problem

• Objective: Need to check the Simple Temporal Network (STN) for temporal consistency and return a conflict – under adding / removing / changing constraints

• Observation: The [partial] plan doesn’t change that much between checks for temporal consistency.

• Approach: Incrementally check for temporal consistency by maintaining data between calls.

I-hsiang Shu, Robert Effinger, Brian Williams, Enabling Fast Flexible Planning through Incremental Temporal Reasoning with Conflict Extraction.
Choosing as a Constraint Satisfaction Problem

The Corresponding Search Tree:
Choosing as a Conditional Constraint Satisfaction Problem

Assign *active* variables A, B, & C such that all of the temporal constraints with true guards are temporally consistent.

*A=1 activates variable B*  
*A=2 activates variable C*
TPN Planning as conflict-directed search

- Decision variables: choices in TPN
- Utility: choice reward (not probability here)
- Consistency check: temporal consistency
Monitoring plans with time

MONITORING TEMPORAL PLANS
Volcano eruption!

What could possibly go wrong??

Base Station

Launch Q

Q fly to Base Station

Q pick up med kit

Q fly to hikers

Q give med kit to climbers
Where do preconditions come from?

Launch Q

Q fly to Base Station

Q pick up med kit

Q fly to hikers

Q give supplies to climbers

Q in the air

Q near med kit

Q has med kit

Q near climbers

Q has empty cargo bay

Q in the air

time
Suppose we have a block-stacking cognitive robot

What sorts of actions might it do?

– Conditions? Effects?
Example PDDL temporal action

(:durative-action pick-up-block)

:parameters (?r - robot ?t ?b - block)

:duration (and (>= ?duration 10) (<= ?duration 30))

:condition (and (at start (clear-above ?t))
             (at start (empty-gripper ?r))
             (over all (can-reach ?r ?t))
             (at start (on ?t ?b)))

:effect (and (at end (not (empty-gripper ?r)))
          (at end (not (on ?t ?b)))
          (at end (holding ?r ?t))
          (at end (clear-above ?b))
          (at end (not (clear-above ?t))))
Temporal plan representation
Events represent points in time
Events may be constrained with simple temporal constraints

\[ e_{s_1} \rightarrow [2, 5] \rightarrow e_{f_1} \]
Actions consist of a start event, end event, and PDDL action

\[ e_{s_1} \quad [2, 5] \quad e_{f_1} \]

(stack-block robot B1 B2)
Temporal plans consist of actions

- $e_{\text{start}}$ to $e_{s_1}$: [0, $\infty$] (put-block-on-ground robot B1)
- $e_{s_1}$ to $e_{f_1}$: [2, 5]
- $e_{f_1}$ to $e_{\text{end}}$: [0, $\infty$] (recharge robot)
- $e_{\text{start}}$ to $e_{\text{end}}$: [3, 15]
- (stack-block robot B1 B2)
PDDL temporal actions have conditions & effects, at start & end

(pick-up-block-from-ground robot B1)

\( e_{s1} \) [10, 30] \( e_{f1} \)

at start condition at start effect at end condition at end effect

maintenance condition
Conditions of actions: effects of prior actions, or come from initial conditions

In the diagram, there are two paths:
- One path labeled "(stack-block robot B1 B2)" with transitions $e_{s1} \rightarrow e_{f1} \rightarrow e_{end}$. The transitions have time intervals $[0, \infty]$.
- Another path labeled "(put-block-on-ground robot B1)" with transitions $e_{start} \rightarrow e_{s1} \rightarrow e_{f1} \rightarrow e_{s1} \rightarrow e_{f1} \rightarrow e_{end}$. The transitions have time intervals $[0, \infty]$, $[2, 5]$, and $[4, 10]$. The path $e_{s1} \rightarrow e_{s1}$ has a time interval $[2, 5]$. The path $e_{f1} \rightarrow e_{f1}$ has a time interval $[2, 5]$. The transition $e_{s1} \rightarrow e_{f1}$ has a time interval $[0, \infty]$. The transition $e_{f1} \rightarrow e_{end}$ has a time interval $[0, \infty]$. The transition $e_{start} \rightarrow e_{s1}$ has a time interval $[0, \infty]$. The transition $e_{end} \rightarrow e_{start}$ has a time interval $[3, 15]$. The transition $e_{end} \rightarrow e_{end}$ has a time interval $[0, \infty]$. The transition $e_{f1} \rightarrow e_{f1}$ has a time interval $[0, \infty]$.
Pike is a *plan executive*: it executes and monitors temporal plans.

- Events must be scheduled + dispatched to robot hardware
- Temporal disturbances handled
- Conditions for success must be monitored
- Signals a problem immediately if detected
Pike problem statement

• **Input:**
  – Temporal plan
  – PDDL operators used in plan
  – Initial state, goal state
  – Stream of state estimates (observations)

• **Output**
  – Dispatch of plan activities at appropriate times
  – Signal if a failure is detected
Pike problem statement

Initial state, goal state

\[ t = 0 : e_{s_1} [2, 5] e_{f_1} \]

(stack-block robot B1 B2)

\[ t = 3 : e_{s_1} [2, 5] e_{f_1} \]

(stack-block robot B1 B2)
Planner-independence: infer relevant monitor conditions from the plan

• Don’t assume monitor conditions are provided by planner

• **Approach**: infer relevant monitor conditions from plan
  – Reason over temporal constraints + conditions
  – Extract *candidate* sets of causal links offline
  – At runtime, monitor relevant causal links during appropriate time interval.
Example of a causal link in temporal plan

(pick-up-block-from-ground robot B1) \( \rightarrow \) (stack-block robot B1 B2)
Causal links can be threatened.

- Suppose that $a_{p_1}$ produces $p$, $a_{p_2}$ produces $\neg p$, and $a_c$ requires $p$ as a condition.
Causal links can be threatened.

- Suppose that $a_{p_1}$ produces $p$, $a_{p_2}$ produces $\neg p$, and $a_c$ requires $p$ as a condition.

Is this OK??
What about this?

- Suppose that $a_{p_1}$ produces $p$, $a_{p_2}$ produces $\neg p$, and $a_c$ requires $p$ as a condition.

NO!! Threatened!
Causal links can dominate each other.

- Now, suppose that $a_{p_1}$ and $a_{p_2}$ both produce $p$, and that $a_c$ requires $p$ as a condition.
However, this domination cannot always be determined before execution.

- Now, suppose that $a_p_1$ and $a_p_2$ both produce $p$, and that $a_c$ requires $p$ as a condition.
But how can we determine these possible execution windows?

- We use an APSP, and examine values with respect to \( e_{\text{start}} \)
Algorithmic approach for execution monitoring of temporally-flexible plans

• (Offline) Extract *candidate* sets of causal links for each consumer, consisting of sets of all mutually non-dominating possible causal links

• (Online) Monitor the order of event execution, and activate one causal link from each candidate set - the “latest” one

• (Online) Continuously check state estimates against all currently-activated causal links
Causal link extraction rough pseudo code (certain details omitted)

• For each consuming event and precondition $p$:
  – Loop over all events and generate set of all mutually incomparable events affecting $p$ that precede consuming event
  – If any events in this set produce $\neg p$ then FAIL
  – Create a new monitor condition for each event in the set
Online causal link monitoring (rough pseudo code)

- While TRUE:
  - Receive current state estimates (from sensors)
  - Check if all activated monitor conditions contained current state, else SIGNAL FAIL
  - Upon dispatching an event, deactivate monitor conditions with event as consuming event. Activate monitor conditions for which no more possible candidates in set remain.
More food for thought...
More food for thought...
More food for thought...
More food for thought...

\[ p \]

require \( p \)

not \( p \)
More food for thought...
More food for thought...