Temporal Plan Execution: Dynamic Scheduling and Simple Temporal Networks

Brian C. Williams
16.412J/6.834J
March 7th, 2005

Outline

• Review: Constraint-based Interval Planning
• Simple Temporal Networks
• Temporal Consistency and Scheduling
• Execution Under Uncertainty

Simple Spacecraft Problem

Observation-1
target
instruments
Observation-2
Observation-3
Observation-4...
pointing
calibrated

Example

Init
Actions
Goal
p_c
p_c

16.410/13: Solved using Graph-based Planners (Blum & Furst)

Partial Order Causal Link Planning (SNLP, UCPOP)

1. Select an open condition
2. Choose an op that can achieve it
   Link to an existing instance
   Add a new instance
3. Resolve threats

Needed Extensions

Time
Resources
Utility
Uncertainty
Representing Timing: Qualitative Temporal Relations [Allen AAAI83]

- A before B
- A meets B
- A overlaps B
- A contains B
- A = B
- A starts B
- A ends B

TakeImage Pictorially

A Temporal Planning Problem

A Consistent Complete Temporal Plan

CBI Planning Algorithm

A Consistent Complete Temporal Plan

Planner Must:
- Check schedulability of candidate plans for correctness.
- Schedule the activities of a complete plan in order to execute.
Relation to Causal Links & Threats

**POCL**
- Causal links:
  - meets
  - proposition

**CBI**
- meets
- proposition

Examples of CBI Planners
- Zeno (Penberthy)
  - intervals, no CSP
- Trains (Allen)
- Descartes (Joslin)
  - extreme least commitment
- iTeT (Ghallab)
  - functional rep.
- HSTS (Muscettola)
  - functional rep., activities
- EUROPA (Jonsson)
  - functional rep., activities
- Kirk (Williams)
  - HTN

Outline

- Review: Constraint-based Interval Planning
- Simple Temporal Networks
- Temporal Consistency and Scheduling
- Execution Under Uncertainty

Qualitative Temporal Constraints

**Metric Time:** Temporal CSPS

(Dechter, Meiri, Pearl 91)

“Bread should be eaten within a day of baking.”

\[ 0 \leq T(\text{baking}) - T(\text{eating}) \leq 1 \text{ day} \]

- \( X_i \)
- \( T_i \)
- \( I_i \) where \( I_i = [a_i, b_i] \)

\[ T_i = (a_i \leq X_i \leq b_i) \text{ or } \ldots \text{ or } (a_i \leq X_i \leq b_i) \]

TCSP Are Visualized Using Directed Constraint Graphs
Simple Temporal Networks (STNs) (Dechter, Meiri, Pearl 91)

At most one interval per constraint
- $T_{ij} = (a_{ij} \leq X_j - X_i \leq b_{ij})$

Sufficient to represent:
- most Allen relations
- simple metric constraints

Can’t represent:
- Disjoint activities

A Temporal Plan Forms an STN

Outline
- Review: Constraint-based Interval Planning
- Simple Temporal Networks
- Temporal Consistency and Scheduling
- Execution Under Uncertainty

TCSP Queries (Dechter, Meiri, Pearl, AIJ91)

- Is the TCSP consistent? Planning
- What are the feasible times for each $X_i$? Planning
- What are the feasible durations between each $X_i$ and $X_j$? Planning
- What is a consistent set of times? Scheduling
- What are the earliest possible times? Scheduling
- What are the latest possible times? Scheduling

To Query an STN, Map to a Distance Graph $G_d = < V, E_d >$

- Edge encodes an upper bound on distance to target from source.
- Negative edges are lower bounds.
- $T_{ij} = (a_{ij} \leq X_j - X_i \leq b_{ij})$
- $X_j - X_i \leq a_{ij}$
- $X_j - X_i \leq b_{ij}$
**G_d Induces Constraints**

- Path constraint: \( i_0 = i, i_1 = \ldots, i_k = j \)
  
  \[ X_j - X_i \leq \sum_{j \neq i} a_{i,j} \]

- Conjoined path constraints result in the shortest path as bound:
  
  \[ X_j - X_i \leq d_{ij} \]

  where \( d_{ij} \) is the shortest path from \( i \) to \( j \)

---

**Conjoined Paths Computed using All Pairs Shortest Path**

(e.g., Floyd-Warshall, Johnson)

1. for \( i := 1 \) to \( n \) do \( d_{ii} := 0 \);
2. for \( i, j := 1 \) to \( n \) do \( d_{ij} := a_{ij} \);
3. for \( k := 1 \) to \( n \) do
   4.   for \( i, j := 1 \) to \( n \) do
      5.     \( d_{ij} := \min\{d_{ij}, d_{ik} + d_{kj}\} \);

---

**Shortest Paths of G_d**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>20</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>-10</td>
<td>0</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>-40</td>
<td>-30</td>
<td>0</td>
<td>-10</td>
</tr>
<tr>
<td>3</td>
<td>-20</td>
<td>-10</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-60</td>
<td>-50</td>
<td>-20</td>
<td>-40</td>
</tr>
</tbody>
</table>

\( d \)-graph

---

**Map To STN Minimum Network**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
<td>50</td>
<td>30</td>
<td>70</td>
</tr>
<tr>
<td>1</td>
<td>-10</td>
<td>0</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>-40</td>
<td>-30</td>
<td>0</td>
<td>-10</td>
</tr>
<tr>
<td>3</td>
<td>-20</td>
<td>-10</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-60</td>
<td>-50</td>
<td>-20</td>
<td>-40</td>
</tr>
</tbody>
</table>

\( d \)-graph

---

**Schedulability: Plan Consistency**

No negative cycles: \( -5 > T_A - T_A = 0 \)

---

**Scheduling: Latest Solution**

Node 0 is the reference.

\( S_i = (d_{0i}, \ldots, d_{ni}) \)

---
Scheduling: Earliest Solution

Node 0 is the reference.

\[ S_i = (-d_{10}, \ldots, -d_{m0}) \]

Scheduling: Window of Feasible Values

Latest Times

\[ \begin{array}{cccc}
0 & 10 & 20 & 30 \\
1 & -10 & 0 & 40 \\
2 & -40 & -30 & 0 \\
3 & -20 & -10 & 20 \\
4 & -60 & -50 & -20 \\
\end{array} \]

Earliest Times

\[ \begin{array}{cccc}
0 & 10 & 20 & 30 \\
1 & 0 & 40 & 60 \\
2 & -30 & 0 & 30 \\
3 & 20 & 0 & 50 \\
4 & -60 & -40 & 0 \\
\end{array} \]

Scheduling: Earliest Solution

• Can assign variables in any order, without backtracking.

\[ \begin{array}{cccc}
0 & 10 & 20 & 30 \\
1 & -10 & 0 & 40 \\
2 & -40 & -30 & 0 \\
3 & -20 & -10 & 0 \\
4 & -60 & -50 & -20 \\
\end{array} \]

• Select value for 1

\[ 15 \quad [10,20] \]

Solution by Decomposition

• Can assign variables in any order, without backtracking.

\[ \begin{array}{cccc}
0 & 10 & 20 & 30 \\
1 & -10 & 0 & 40 \\
2 & -40 & -30 & 0 \\
3 & -20 & -10 & 0 \\
4 & -60 & -50 & -20 \\
\end{array} \]

• Select value for 1

\[ 15 \quad [10,20] \]

Solution by Decomposition

• Can assign variables in any order, without backtracking.

\[ \begin{array}{cccc}
0 & 10 & 20 & 30 \\
1 & -10 & 0 & 40 \\
2 & -40 & -30 & 0 \\
3 & -20 & -10 & 0 \\
4 & -60 & -50 & -20 \\
\end{array} \]

• Select value for 2, consistent with 1

\[ 45 \quad [40,50], [30,40] \]
Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of \( Y \) using all selected \( X \): \( Y \in X + |XY| \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 20 50 30 70</td>
<td></td>
</tr>
<tr>
<td>1 -10 0 40 20 60</td>
<td></td>
</tr>
<tr>
<td>2 -40 -30 0 -10 30</td>
<td></td>
</tr>
<tr>
<td>3 -20 -10 20 0 50</td>
<td></td>
</tr>
<tr>
<td>4 -60 -50 -20 -40 0</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow 15 \)

- Select value for 1

- Select value for 2, consistent with 1

\( \Rightarrow 45 \) \( [45,50] \)

\( d \)-graph

Outline

- Review: Constraint-based Interval Planning
- Simple Temporal Networks
- Temporal Consistency and Scheduling
- Execution Under Uncertainty

\( O(N^2) \)
Executing Flexible Temporal Plans
[Muscettola, Morris, Pell et al.]

Handling delays and fluctuations in task duration:
• Least commitment temporal plans leave room to adapt.

Flexible execution adapts through dynamic scheduling.
• Assigns time to event when executed.

Issues in Flexible Execution

1. How do we minimize execution latency?

2. How do we schedule at execution time?
Issues in Flexible Execution

1. How do we minimize execution latency?
   ➔ Propagate through a small set of neighboring constraints.
2. How do we schedule at execution time?
Issues in Flexible Execution

1. How do we minimize execution latency?
   ➔ Propagate through a small set of neighboring constraints.

2. How do we schedule at execution time?

Dynamic Scheduling by Decomposition

- Compute APSP graph
- Decomposition enables assignment without search

Assignment by Decomposition

- Select executable timepoint and assign
- Propagate assignment to neighbors
Assignment by Decomposition
- Select executable timepoint and assign
- Propagate assignment to neighbors

Solution:
- Assignments must monotonically increase in value.
- First execute all APSF neighbors with negative delays.

But C now has to be executed at t = 2, which is already in the past!

Dispatching Execution Controller
Execute an event when enabled and active
- Enabled - APSP Predecessors are completed
  - Predecessor – a destination of a negative edge that starts at event.
- Active – Current time within bound of task.

Dispatching Execution Controller
Initially:
- E = Time points w/o predecessors
- S = {}
Repeat:
1. Wait until current_time has advanced so:
   a. Some TP in E is active
   b. All time points in E are still enabled.
2. Set TP's execution time to current_time.
3. Add TP to S.
4. Propagate time of execution to TP’s APSF immediate neighbors.
5. Add to A, all immediate neighbors that became enabled.
   a. TPs enabled if all negative edges starting atTP’s have their destination in S.

Propagation is Focused
- Propagate forward along positive edges to tighten upper bounds.
  - forward prop along negative edges is useless.
- Propagate backward along negative edges to tighten lower bounds.
  - Backward prop along positive edges useless.

Propagation Example
S = {A}

Propagation Example
S = {A}
Propagation Example

$$S = \{A\}$$

Reducing Execution Latency

Filtering:
- some edges are redundant
- remove redundant edges

Execution time is:
- worst case $O(n)$
- best case $O(n)$

Edge Domination

- BC upper-dominates AC if in every consistent execution, $T_B + b(B,C) \leq T_A + b(A,C)$

- The thread running through A-B-C is always just as fast or faster than the thread running through A-C

Edge Domination

- AB lower-dominates AC if in every consistent execution, $T_B - b(A,B) \geq T_C - b(A,C)$

- Enablement of node A is always determined by thread running through A-B-C
• Edge Dominance
  – Eliminate edge that is redundant due to the triangle inequality $AB + BC = AC$

An Example of Edge Filtering

• Start off with the APSP network

• Start at A-B-C triangle
An Example of Edge Filtering

• Look at B-D-C triangle

• Look at D-A-B triangle

• Look at D-A-C triangle

• Look at B-C-D triangle
### An Example of Edge Filtering

- Resulting network has less edges than the original

```
A 9 1 B 1 D
          10
0        0 1 2 2
```

### Additional Filtering

- Node Contraction
  - Collapse two events with fixed time between them

```
A

0
```

### Additional Filtering

- Node Contraction
  - Collapse two events with fixed time between them

```
A

0
```

### An Example of Node Contraction

- Resulting network has less edges than the original

```
A 0 B

0
```

### Avoiding Intermediate Graph Explosion

**Problem:**
- APSP consumes $O(n^2)$ space.

**Solution:**
- Interleave process of APSP construction with edge elimination
  - Never have to build whole APSP graph

**Goals and Environment Constraints**

- Temporal Planner
- Temporal Network Solver
- Dynamic Scheduling and Task Execution
- Model-based Programming & Task Execution
- Observations
- Commands

- Projective Task Expansion
- Reactive Task Expansion
- Temporal Plan
- Modes
- Goals
- Task Dispatch