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

Example

Init Actions Goal

\[ \begin{align*}
\text{Im} \quad p_C \quad c \quad \text{Im} \\
\text{px} \quad p_Y \quad \text{Im} \\
c \quad \text{Im} \quad l_a
\end{align*} \]

Example

\[ \begin{align*}
\text{Im} \quad p_C \quad c \\
\text{Im} \quad p_Y \\
c \quad \text{Im} \quad l_a
\end{align*} \]

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

A Consistent Complete Temporal Plan

CBI Planning Algorithm

Choose:
- introduce an action & instantiate constraints
- coalesce propositions
- Propagate temporal constraints

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: action \( \rightarrow \) proposition

**CBI**
- Threats: action \( \rightarrow \) proposition

### Examples of CBI Planners

- **Zeno** (Penberthy): intervals, no CSP
- **Descartes** (Joslin): extreme least commitment
- **lxTeT** (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

**Maybe Expressed as Inequalities**

- \( x \text{ before } y \): \( X^+ < Y^- \)
- \( x \text{ meets } y \): \( X^+ = Y^- \)
- \( x \text{ overlaps } y \): \( (Y^- < X^+) \land (X^- < Y^+) \)
- \( x \text{ during } y \): \( (Y^- < X^-) \land (X^+ < Y^+) \)
- \( x \text{ finishes } y \): \( (X^- < Y^-) \land (X^+ = Y^+) \)
- \( x \text{ equals } y \): \( (X^- = Y^-) \land (X^+ = Y^+) \)

Inequalities may be expressed as binary interval relations:

\[ Y^- < X^- < 0, +\infty \]

Generalize to include metric constraints:

\[ Y^- - \infty < [l_b, u_b] \]

### Metric Time: Temporal CSPS

(Dechter, Meiri, Pearl 91)

- “Bread should be eaten within a day of baking.”
- \( 0 \leq [T(\text{baking}) - T(\text{eating})] \leq 1 \text{ day} \)

\[ < X_i, T_i, T_j > \]

- \( X_i \) continuous variables
- \( T_i, T_j \) interval constraints
- \( \{I_1, \ldots, I_k\} \) where \( I_i = [a, b_i] \)

- \( T_i = (a_i \leq X \leq b_i) \) or \( \ldots \) or \( (a_k \leq X \leq b_k) \)
- \( T_j = (a_k \leq X - X_i \leq b_k) \) or \( \ldots \) or \( (a_k \leq X - X_i \leq b_k) \)

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

At most one interval per constraint

- $T_{ij} = (a_{ij} \leq X_i - X_j \leq b_{ij})$

Can’t represent:
- Disjoint activities

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

Outline

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

To Query an STN, Map to a Distance Graph $G_d = \langle V, E_d \rangle$

- Edge encodes an upper bound on distance to target from source.
- Negative edges are lower bounds.

TCSP Queries  
(Dechter, Meiri, Pearl, Al91)

- 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

A Temporal Plan Forms an STN

Outline

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

To Query an STN, Map to a Distance Graph $G_d = \langle V, E_d \rangle$

- Edge encodes an upper bound on distance to target from source.
- Negative edges are lower bounds.

TCSP Queries  
(Dechter, Meiri, Pearl, Al91)

- 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

A Temporal Plan Forms an STN

Outline

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

To Query an STN, Map to a Distance Graph $G_d = \langle V, E_d \rangle$

- Edge encodes an upper bound on distance to target from source.
- Negative edges are lower bounds.

TCSP Queries  
(Dechter, Meiri, Pearl, Al91)

- 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
Gd Induces Constraints

- Path constraint: \( i_0 = i, i_1 = \ldots, i_k = j \)
  \[ X_j - X_i \leq \sum_{j=1}^{i} a_{i,j} \]
- 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 \( d_{ii} \leftarrow 0 \);
2. for \( i, j := 1 \) to \( n \) do \( d_{ij} \leftarrow a_{ij} \);
3. for \( k := 1 \) to \( n \) do \( d_{ij} \leftarrow \min(d_{ij}, d_{ik} + d_{kj}) \);

Shortest Paths of \( G_d \)

Map To STN Minimum Network

Schedulability: Plan Consistency

No negative cycles: \(-5 > T_{f} - T_{f} = 0\)

Scheduling: Latest Solution

Node 0 is the reference.

\[ S_t = (d_{01}, \ldots, d_{0n}) \]
Scheduling: Earliest Solution

Node 0 is the reference. 

\[ S_1 = (-d_1, \ldots, -d_n) \]

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 \\
0 & 0 & 20 & 30 & 70 \\
1 & -10 & 0 & 20 & 60 \\
2 & -40 & -30 & 0 & -10 \\
3 & -20 & -10 & 0 & 50 \\
4 & -60 & -50 & -20 & -40
\end{array} \]

\[ d\text{-graph} \]

Scheduling: Window of Feasible Values

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
0 & 0 & 20 & 30 & 70 \\
1 & -10 & 0 & 40 & 20 & 60 \\
2 & -40 & -30 & 0 & -10 & 30 \\
3 & -20 & -10 & 0 & 50 & 0 \\
4 & -60 & -50 & -20 & -40 & 0
\end{array} \]

Latest Times

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

Earliest Times \[ d\text{-graph} \]

Scheduling without Search: Solution by Decomposition

• Can assign variables in any order, without backtracking.

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
0 & 0 & 20 & 30 & 70 \\
1 & -10 & 0 & 20 & 60 \\
2 & -40 & -30 & 0 & -10 \\
3 & -20 & -10 & 0 & 50 \\
4 & -60 & -50 & -20 & -40
\end{array} \]

• Select value for 1

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

\[ d\text{-graph} \]

Solution by Decomposition

• Can assign variables in any order, without backtracking.

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
0 & 0 & 20 & 30 & 70 \\
1 & -10 & 0 & 40 & 20 \\
2 & -40 & -30 & 0 & -10 \\
3 & -20 & -10 & 0 & 50 \\
4 & -60 & -50 & -20 & -40
\end{array} \]

• Select value for 1

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

\[ d\text{-graph} \]

Solution by Decomposition

• Can assign variables in any order, without backtracking.

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
0 & 0 & 20 & 30 & 70 \\
1 & -10 & 0 & 40 & 20 \\
2 & -40 & -30 & 0 & -10 \\
3 & -20 & -10 & 0 & 50 \\
4 & -60 & -50 & -20 & -40
\end{array} \]

• Select value for 1

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

\[ d\text{-graph} \]

Solution by Decomposition

• Can assign variables in any order, without backtracking.

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
0 & 0 & 20 & 30 & 70 \\
1 & -10 & 0 & 40 & 20 \\
2 & -40 & -30 & 0 & -10 \\
3 & -20 & -10 & 0 & 50 \\
4 & -60 & -50 & -20 & -40
\end{array} \]

• Select value for 2, consistent with 1

\[ \rightarrow 45 \quad [40, 50], [40, 40] \]

\[ d\text{-graph} \]
### 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></th>
<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>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-10</td>
<td>0</td>
<td>40</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>-40</td>
<td>-30</td>
<td>0</td>
<td>-10</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>-20</td>
<td>-10</td>
<td>20</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>-60</td>
<td>-50</td>
<td>-20</td>
<td>-40</td>
<td>0</td>
</tr>
</tbody>
</table>

$\Rightarrow 15$

- Select value for $1$
- Select value for 2, consistent with 1
- Select value for $3$, consistent with 1 & 2
- Select value for 4, consistent with 1, 2 & 3

### Outline

- Review: Constraint-based Interval Planning
- Simple Temporal Networks
- Temporal Consistency and Scheduling
- Execution Under Uncertainty
Executing Flexible Temporal Plans
[Muscettola, Morris, Pell et al.]
Handling delays and fluctuations in task duration:
• Least commitment temporal plans leave room to adapt.

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

Time Propagation Can Be Costly
Time Propagation Can Be Costly

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?

Compile to Efficient Network
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

Equivalent All Pairs Shortest Paths (APSP) Representation
Assignment by Decomposition

• Select executable timepoint and assign
• Propagate assignment to neighbors

Solution:
• Assignments must monotonically increase in value.
  ➔ First execute all APSP neighbors with negative delays.

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 s
   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 APSP immediate neighbors.
5. Add to A, all immediate neighbors that became enabled.
   a. TP’s 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}

S = {A}

Propagation Example

S = {A}
E = { C }

Reduction Execution Latency

Filtering:
• some edges are redundant
• remove redundant edges

Execution time is:
• worst case O(n)
• best case O(1)

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

An Example of Edge Filtering

• 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 B-C-D 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

![Graph](image)

Additional Filtering

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

![Graph](image)

Additional Filtering

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

![Graph](image)

An Example of Node Contraction

- Resulting network has less edges than the original

![Graph](image)

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