KNOWLEDGE OF SPACE AND TIME

NICK HAWES, SCHOOL OF COMPUTER SCIENCE, UNIVERSITY OF BIRMINGHAM

N.A.HAWES@CS.BHAM.AC.UK
HTTP://WWW.CS.BHAM.AC.UK/~NAH
What is the problem? Giving robots an understanding of space and time.

Why is it hard? Finding the right abstraction for the right problem, avoid complexity yet providing reasoning power.
G4S Technology, UK

Haus der Barmherzigkeit, Austria
Haus der Barmherzigkeit, Vienna, Austria

$1030\, m^3 \quad 17 \text{ Nodes}$
G4S Technology, Challenge House, Tewkesbury, UK
690m³  46 Nodes
Task Scheduling for Mobile Robots Using Interval Algebra
Mudrová and Hawes. In, ICRA ’15.

How to tell a robot what time to do something?

Not just order, but precise starting times (e.g. 14:02)

Considering up to 100 tasks
∀i : \( \min \sum (t_i - s_i) \)
Coltin et al.*

Scheduling using mixed-integer programming

\[ S_i \leq t_i \land (t_i + d_i) \leq e_i \]

Coltin et al.* Scheduling using mixed-integer programming

\[ S_i \leq t_i \land (t_i + d_i) \leq e_i \]

\[ \forall i,j : t_i + d_i + time(p^e_i,p^s_j) \leq t_j \]

or

\[ \forall i,j : t_j + d_j + time(p^e_j,p^s_i) \leq t_i \]

\[ \forall i : min \sum (t_i - s_i) \]

∀ i, j : \( t_i + d_i + time(p^e_i, p^s_j) \leq t_j \)

or

∀ i, j : \( t_j + d_j + time(p^e_j, p^s_i) \leq t_i \)
Reasoning about **tasks’ time windows**: $S_i E_i$ vs $S_j E_j$

<table>
<thead>
<tr>
<th>Constraint</th>
<th>$i$ before $j$</th>
<th>$i$ overlaps $j$</th>
<th>$j$ overlaps $i$</th>
<th>$i$ during $j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$ before $j$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$i$ meets $j$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$j$ before $i$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$j$ meets $i$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Allen’s Interval Algebra**

- **i before j**
  - no order constraint

- **i overlaps j**
  - choose only possible order constraint

  
  ![Diagram showing i overlaps j]

  

- **i overlaps j**
  - choose order to satisfy \( \forall i : \min \sum (t_i - s_i) \)

- **i equals j**
  - pick first one seen
<table>
<thead>
<tr>
<th></th>
<th>Care</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td># Problems</td>
<td>606</td>
<td>358</td>
</tr>
<tr>
<td>Smallest Problem</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Largest Problem</td>
<td>135</td>
<td>71</td>
</tr>
<tr>
<td>Mean Problem Size</td>
<td>28.88 ($\sigma$ 26.28)</td>
<td>9.59 ($\sigma$ 12.97)</td>
</tr>
<tr>
<td># Problems &gt;15</td>
<td>349 (58%)</td>
<td>106 (30%)</td>
</tr>
</tbody>
</table>
Task

\[ t_i \]

action

\[ d_i \]

\[ p_{s_i} \]

\[ p_{e_i} \]
Optimal and Dynamic Planning for Markov Decision Processes with Co-Safe LTL Specifications
Lacerda, Parker and Hawes. In, IROS’14.
Best 8 matches between straight-line and recorded times
Worst 8 matches between straight-line and recorded times
**action** goto W2 from W1

**cost** mean time from all attempts
Goal is to be in state \( W3 \)

Policy:

- \( W1 \) to \( W3 \)
- \( W2 \) to \( W3 \)
- \( W3 \) to \( W3 \)

Diagram:

- From \( W1 \) to \( W3 \) with weight 3
- From \( W2 \) to \( W3 \) with weight 3
- From \( W3 \) to \( W1 \) with weight 5
Eventually reach $W_2$ and $W_3$

Policy:

1. $W_1 \rightarrow W_3$
2. $W_2 \rightarrow W_3$
3. $W_3 \rightarrow W_3$

$(F \ W_2) \land (F \ W_3)$
\[(F \ W2) \land (F \ W3)\]

Eventually reach \(W2\) and \(W3\)

Cool tool: [http://www.lsv.ens-cachan.fr/~gastin/ltl2ba](http://www.lsv.ens-cachan.fr/~gastin/ltl2ba)
"init"
The resulting policy provides the optimal action to take in any state.

The LTL element provides memory to the policy.

A side effect of producing the policy is expected costs (time) for reaching states.

These are passed to the scheduler.
What is the problem? Giving robots an understanding of space and time

Why is it hard? Finding the right abstraction for the right problem, avoid complexity yet providing reasoning power