Robust Execution of Contingent, Temporally Flexible Plans

Stephen Block
Andreas Wehowsky, Brian Williams

Real World Autonomous Agents

- Coordinate multiple agents
- Provide robustness

Robustness to Disturbances

Robustness to...
- Temporal uncertainty: Temporally flexible mission plan
- Execution uncertainty: Dispatchable execution
- Communication latency: Distributed architecture
- Plan failure: Contingent mission plan

Robustness to Temporal Uncertainty

Robustness to...
- Temporal uncertainty: Temporally flexible mission plan

Temporally flexible plans allow activities with uncertain duration

“Drive to rock”

Hardware command

Robustness to Execution Uncertainty

Robustness to...
- Temporal uncertainty: Temporally flexible mission plan
- Execution uncertainty: Dispatchable execution

Problem: Plan is either...  
- Brittle to temporal execution uncertainty
- Overly conservative to ensure success

Solution: Dispatchable execution...
- Postpone scheduling until execution time

Maintenance of temporal flexibility

Reactively schedule execution times

Hardware Commands
Robustness to Execution Uncertainty

Least commitment planning allows the executive to use temporal flexibility to respond to uncertainties at run time.

Robustness to Communication Latency

Problem: Centralized architecture introduces communication bottleneck at master agent.

Solution: A distributed architecture even out the communication requirements.

Distributed Architecture

Reduced computational complexity.

Avoids communication bottleneck.
Plan Distribution

- Leader election

Robustness to Plan Failure

Robustness to:
- Temporal uncertainty: Temporally flexible mission plan
- Execution uncertainty: Dispatchable execution
- Communication latency: Distributed architecture
- Plan failure: Contingent mission plan

Distributed Architecture

- Reduced computational complexity
  - Plan distribution
  - Plan selection
- Execution:
  - Temporally flexible plan
  - Contingent temporally flexible plan
- Avoids communication bottleneck
- Reduced computational complexity

Interleaved Candidate Generation and Consistency Checking

1. Generate candidate plans through distributed search on the TPN
2. Test the generated plans for temporal consistency

Candidate Generation

1. Generate candidate plans through distributed search on the TPN
2. Test the generated plans for temporal consistency

Implemented using a message passing scheme...

- findfirst: Initial search for a consistent set of choice variable assignments
- findnext: Search for a new consistent assignment, to achieve global consistency
- fail: No consistent set of choice variable assignments was found
- ack: A consistent set of choice variable assignments was found
Candidate Generation

Nodes send findfirst messages to their children
Search progresses at increasing depth

Consistency Checking

1. Generate candidate plans through distributed search on the TPN
2. Test the generated plans for temporal consistency

Interleaved and concurrent

Implemented with synchronized distributed Bellman Ford Single Source Shortest Path algorithm

- Requires only local knowledge
- Uses a message passing scheme
- Runs on distance graph corresponding to active portions of the TPN
- Processor synchronization at every update cycle gives linear run time

Consistency Checking

Nodes perform consistency checking and report to their parent
Checking progresses at decreasing depth

Candidate Generation

Nodes send findfirst messages to their children
Search progresses at increasing depth

Time Complexity Analysis

<table>
<thead>
<tr>
<th></th>
<th>Centralized Planner</th>
<th>DTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate Generation</td>
<td>N^e</td>
<td>N^e</td>
</tr>
<tr>
<td>Temporal Consistency Checking</td>
<td>N\log N + NM</td>
<td>N</td>
</tr>
<tr>
<td>Overall Time Complexity</td>
<td>Exponential</td>
<td>Exponential</td>
</tr>
</tbody>
</table>

N Number of nodes
M Number of edges
e Size of domain of choice variables

- Worst case complexity of candidate generation corresponds to a plan entirely composed of choice nodes
- DTP uses parallel processing for parallel and sequence networks
- Time complexity much lower in realistic examples
- In practice, complexity of DTP is near-linear
Questions?