Hierarchical Learning for Human-Robot Collaboration

Brian Scassellati
scaz@cs.yale.edu
http://scazlab.yale.edu
Sequential Manipulation Tasks
SMDPs Modeling Tasks Are Complex
SMDPs Modeling Tasks Are Complex

There’s an execution policy in here somewhere...
Near-Term Assistive Scenarios
Changes as we consider Collaborative Manufacturing

• Keep many of our pillars
  – Observations of sequential manipulation tasks
  – Existing methods for learning from demonstration
  – Focus on execution policies

• Adapt to collaborative setting
  – Move away from flat representations
  – Move away from divide-and-conquer planning mechanisms
  – Consider collaboration in a broad sense
Changes as we consider Collaborative Manufacturing

• Keep many of our pillars
  – Observations of sequential manipulation tasks
  – Existing methods for learning from demonstration
  – Focus on execution policies

• Adapt to collaborative setting
  – Move away from flat representations
  – Move away from divide-and-conquer planning mechanisms
  – Consider collaboration in a broad sense
Benefits of Hierarchical Structure

• Benefits of hierarchical structure:
  – Reduce dimensionality of policy search space
Hierarchical structure within the SMDP can be used to make the policy search more tractable.
Leverage Hierarchical Structure

Hierarchical structure within the SMDP can be used to make the policy search more tractable.
Benefits of Hierarchical Structure

• Benefits of hierarchical structure:
  – Reduce dimensionality of policy search space
  – Multi-agent task allocation
    • Parallel execution
    • Preferential allocation
Augmenting Hierarchical Plans with Social Metadata

Required Roles

Robot Only
Human Only
Either
Requires Both
Mixed

Task
(G | E) -> H

Sequencing and Ordering Constraints

Resource Requirements (tools)

Timing (per agent)

Human: 10 sec
Benefits of Hierarchical Structure

• Benefits of hierarchical structure:
  – Reduce dimensionality of policy search space
  – Multi-agent task allocation
    • Parallel execution
    • Preferential allocation
  – Transparency
    • Similarity of cognitive models
    • Ability to leverage communication
Benefits of Hierarchical Structure

• Benefits of hierarchical structure:
  – Reduce dimensionality of policy search space
  – Multi-agent task allocation
    • Parallel execution
    • Preferential allocation
  – Transparency
    • Similarity of cognitive models
    • Ability to leverage communication

• Disadvantages:
  – How do we build the hierarchy from observation?
Building Hierarchical Structure from Sequential Observations

Performance by worker “A”

Performance by worker “B”

Upgrade Laptop RAM

Prepare RAM
- Unpackage RAM
- Place RAM in workspace
- Orient laptop
- Open memory housing
- Remove old RAM
- Place old RAM in recycling area
- Insert new RAM
- Place old RAM in recycling area
- Replace memory housing
- Orient laptop

Prepare Laptop
- Unpackage RAM
- Place RAM in workspace
- Orient laptop
- Open memory housing
- Remove old RAM
- Place old RAM in recycling area
- Insert new RAM
- Place old RAM in recycling area
- Replace memory housing
- Orient laptop

Replace RAM
- Unpackage RAM
- Place RAM in workspace
- Orient laptop
- Open memory housing
- Remove old RAM
- Place old RAM in recycling area
- Insert new RAM
- Place old RAM in recycling area
- Replace memory housing
- Orient laptop

Finalize Laptop
- Unpackage RAM
- Place RAM in workspace
- Orient laptop
- Open memory housing
- Remove old RAM
- Place old RAM in recycling area
- Insert new RAM
- Place old RAM in recycling area
- Replace memory housing
- Orient laptop
SMDP of “Attach Front Frame” Subtask

(Hayes & Scassellati, IROS 2014)
SMDP-Conjugate of “Attach Front Frame” Subtask

SMDP Conjugate: Actions become vertices and required state is described on edges as a composition of motor primitives.

Edges are labeled with transition requirements – A composition of motor primitives describing the world state required to use that edge.

Vertices contain motor primitives that can be executed only upon arriving in the node.
Building Hierarchical Structure

Task Hierarchy

Get L.Peg

Place L.Peg

Get R.Peg

Place R.Peg

Place L.Peg

Get L.Peg

Place R.Peg

Get R.Peg

Get Frame

Place Frame

State

Action

Goal

Step 1: Find Cliques (0)
Building Hierarchical Structure

State
Action
Goal

Step 2: Find Chains (2)

Task Hierarchy

Get L.Peg → Place L.Peg

Get R.Peg → Place R.Peg

Get Frame → Place Frame

Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg

Get Frame → Place Frame → Get Frame → Place L.Peg → Have L.Peg → Place R.Peg → Get R.Peg
Building Hierarchical Structure

State
Action
Goal

Task Hierarchy

Step 1: Find Cliques (1)
Building Hierarchical Structure

Task Hierarchy

Step 2: Find Chains (1)
Changes as we consider Collaborative Manufacturing

• Keep many of our pillars
  – Observations of sequential manipulation tasks
  – Existing methods for learning from demonstration
  – Focus on execution policies

• Adapt to collaborative setting
  – Move away from flat representations
  – Move away from divide-and-conquer planning mechanisms
  – Consider collaboration in a broad sense
Supportive Behaviors
Can we do better than LfD-based Methods?

Demonstration-based Methods

- **Human** figures out *how and when* the robot can be helpful
  - Quickly enables useful, helpful actions.
  - Does not scale with task count!
  - Requires human expert

Planner-based Methods

- **Robot** figures out *how and when* it can be helpful
  - Allows for novel behaviors to be discovered
  - Enables deeper task comprehension and action understanding

Can we do better than LfD-based Methods?
Generating Supportive Behaviors

Perspective Taking  Symbolic planning  Motion planning

Autonomously Generated Supportive Behaviors
Supportive Behavior Planning

Hypothetical Environment Generator

Current State

Goal Predicates

Current State

Initial State

Goal State

Support Agent Planner

Lead Agent Planner Model

Initial State

Goal State

Policies

Policy Weighting Function

Multi-Agent Plan Evaluation

Support Policy
Supportive Behavior Planning

- Hypothesize future world states based on their plans
Supportive Behavior Planning

- Hypothesize future world states based on their plans
- Predict lead agent behavior using a user model
Supportive Behavior Planning

- Hypothesize future world states based on their plans
- Predict lead agent behavior using a user model
- Plan supportive actions that would simplify achieving this world state (or prevent sub-optimal plans)
Supportive Behavior Planning

- Hypothesize future world states based on their plans
- Predict lead agent behavior using a user model
- Plan supportive actions that would simplify achieving this world state (or prevent sub-optimal plans)
- Evaluate multi-agent plan
Simplified manipulation task
Supportive Action for Bench Assembly
Simplified Vision, Control
Failure Recovery

Right Arm
State: START

Left Arm
State: START
Preferences in Task Assignment
Changes as we consider Collaborative Manufacturing

• Keep many of our pillars
  – Observations of sequential manipulation tasks
  – Existing methods for learning from demonstration
  – Focus on execution policies

• Adapt to collaborative setting
  – Move away from flat representations
  – Move away from divide-and-conquer planning mechanisms
  – Consider collaboration in a broad sense
Application Domain: SnapCircuits

- “Construct a switched circuit with a power source and an LED”
- Many valid solutions
- Many suboptimal solutions exist
- Rigid, easily identified components
Application Domain: SnapCircuits

- Resource utilization
- Circuit board space utilization
- Role assignment
- Subtask parallelization
Supportive Behavior Planning

- Hypothesize future world states based on their plans
- Predict lead agent behavior using a user model
- Plan supportive actions that would simplify achieving this world state (or prevent sub-optimal plans)
- Evaluate multi-agent plan
Plan Evaluation

Choose the support policy ($\xi \in \Xi$) that minimizes the expected execution duration of the leader’s policy ($\pi \in \Pi$) to solve the TAMP problem $T$ from the current state ($s_c$)

- Duration estimate must account for
  - Resource conflicts (shared utilization/demand)
  - Spatial constraints (support agent’s avoidance of lead)

$$\min_{\xi \in \Xi} \sum_{\pi \in \Pi_T} w_\pi \ast \text{duration}(T, \pi, \xi, s_c, \gamma)$$
Plan Evaluation

Choose the support policy ($\xi \in \Xi$) that minimizes the expected execution duration of the leader’s policy ($\pi \in \Pi$) to solve the TAMP problem $T$ from the current state ($s_c$)

- Duration estimate
  - Resource conflicts (shared utilization/demand)
  - Spatial constraints (support agent’s avoidance of lead)

$$\min_{\xi \in \Xi} \sum_{\pi \in \Pi_T} w_\pi \ast \text{duration}(T, \pi, \xi, s_c, \gamma)$$

Weighting function makes a big difference!
Uniform Weighting Functions \( \omega_\pi = 1 \)
Optimality-Proportional Weighting

\[ w_\pi = \left( \frac{\min_{\pi \in \Pi_T} \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1)}{\text{duration}(T, \pi, \emptyset, s_0, f(x) = 1)} \right)^p \]

Weight plans proportional to similarity vs. the best-known solution
Optimality-Proportional Weighting
Error Mitigation Weighting

\[ w_\pi = \begin{cases} 
  f(\pi) & ; \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1) \leq \epsilon \\
  - \alpha w_\pi & ; \text{otherwise}
\end{cases} \]

Plans more optimal than some cutoff \( \epsilon \) are treated normally, per \( f \).

Suboptimal plans are negatively weighted, encouraging active mitigation behavior from the supportive robot.

\[ \alpha \leq \frac{1}{\max_{\pi} w_\pi} \]

is a normalization term to avoid harm due to plan overlap.
Error Mitigation Weighting
People who did all the work

Brad Hayes
Alessandro Roncone
Olivier Mangin
Francesca Stramandinoli
Thanks to...