Multi-Level Learning in Hybrid Deliberative/Reactive Mobile Robot Architectural Software Systems

Georgia Tech College of Computing
Georgia Tech Research Institute
Mobile Intelligence Corporation
March 2001
Summary of Approach

• Investigate robot shaping at five distinct levels in a hybrid robot software architecture
• Implement algorithms within *MissionLab* mission specification system
• Conduct experiments to evaluate performance of each technique
• Combine techniques where possible
• Integrate on a platform more suitable for realistic missions and continue development
Overview of techniques

- **CBR Wizardry**
  - Guide the operator

- **Probabilistic Planning**
  - Manage complexity for the operator

- **RL for Behavioral Assemblage Selection**
  - Learn what works for the robot

- **CBR for Behavior Transitions**
  - Adapt to situations the robot can recognize

- **Learning Momentum**
  - Vary robot parameters in real time

THE LEARNING CONTINUUM:

- **Deliberative (premission)**
- **Behavioral switching**
- **Reactive (online adaptation)**
Learning Momentum

• Behavioral parameters are modified at runtime depending on a robot’s prior success in navigating the environment

• Robot stores a short history of items such as the number of obstacles encountered, the distance to the goal, and other relevant data
  – uses this history to determine which one of several predefined situations the robot is in and alters its behavioral gains accordingly
  – a crude form of reinforcement learning, where if the robot is doing well, it should keep doing what it's doing and even do it more

• Two strategies were previously described: ballooning and squeezing
Learning Momentum Trials

- Augments earlier simulation trials with real robot results
- In a limited number of runs, success was achieved only with LM active
- Relied on sonar sensing of obstacles
- Future experiments will use laser scanners on indoor/outdoor robot
- Recent effort addresses integration with CBR learning

![Graph showing average steps to completion for a real environment. Trials with no successful runs were given the largest value on the graph.](image)
CBR for Behavioral Selection

- As the environment of a reactive robot changes, the selected behaviors should change.
- Previous results showed improvements in simulation.
Behavioral Adaptation Approach

- Select behavioral assemblages based on robot mission specification
- Adjust parameters with CBR techniques
- Fine-tuning the behaviors allows the library of cases to remain smaller
  - Initially done only once, based on temporal “progress” measure
  - Now a continuous process, integrating with Learning Momentum method
CBR Trials

- Ten runs were conducted with the CBR module and ten without
- Test course included box canyon and obstacle field
- Obstacle density varied from low to high
- Results correlate well with the simulation-based data
  - as the average obstacle density increases, the benefits from the CBR module also increase.

<table>
<thead>
<tr>
<th>Obstacle Density</th>
<th>Traveled Distance</th>
<th>Time Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>6.2</td>
<td>3.3</td>
</tr>
<tr>
<td>Medium</td>
<td>17.8</td>
<td>17.6</td>
</tr>
<tr>
<td>High</td>
<td>26.4</td>
<td>28.6</td>
</tr>
</tbody>
</table>
Integration of LM and CBR

- The first of several integration steps to combine the advantages of different robot shaping methods
- CBR module provides discontinuous switching of behavioral parameters based on sufficiently drastic changes in the environment
- LM module provides a continuous adaptation of behavioral parameters
Specifics of LM/CBR Integration

• CBR module selects a new case either
  – when environment characteristics significantly change, or
  – when robot performance falls below a threshold for a specified interval

• A case is defined as before, but now includes a set of parameters that control the LM adaptation rules

• LM Module acts as before, but is “conditioned” by the CBR-provided parameters

• The previous library of cases is insufficient
  – Lacks adaptation parameters
  – Does not address outdoor environments
  – Larger parameter space will make manual case building difficult and time-consuming
Automatic Case Learning

- Addresses the rebuilding of the case library
- CBR library now contains cases with both positive and negative performance history
- Reinforcement function computation sub-module computes a reinforcement function which is used to adjust the performance history measure for the last K applied cases
- The previous random selection is now weighted by the goodness of each of the spatially and temporally matching cases
CBR “Wizardry”

• Help the user during mission specification
  – check for common mistakes
  – suggest fixes
  – automatically insert elements

• Previous highlights include the addition of a plan creation recorder and initial usability studies
Usability studies

• Conducted a set of experiments
  – to evaluate the usability of the MissionLab interface
  – to determine to what extent the current interface enables novice users to design relevant missions
  – to provide a baseline against which future versions of the system will be evaluated
Test subject demographics

- Which Operating Systems Are You Familiar With?
- Mouse Experience
- How Long Have You Been Programming?
- Do you Like Computer Games?
- How good are you at giving directions?
- Military Experience
Usability study results

- Results suggest that novices perform nearly as well as experienced users
- Two-robot scenario was considerably more difficult than single-robot scenario
- Studies have contributed to the population of a case database that will be used in the initial implementation of the wizard
- Summary data for all subjects:

<table>
<thead>
<tr>
<th></th>
<th>Single Robot Scenario</th>
<th>Two Robot Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Tasks</td>
<td>14.23</td>
<td>23.40 (both robots)</td>
</tr>
<tr>
<td>Number of Triggers</td>
<td>21.27</td>
<td>38.50 (both robots)</td>
</tr>
<tr>
<td>Modifications</td>
<td>36.32</td>
<td>69.00</td>
</tr>
<tr>
<td>Modification Time</td>
<td>6 min 10 sec</td>
<td>12 min 11 sec</td>
</tr>
<tr>
<td>Total Completion Time</td>
<td>33 min 3 sec</td>
<td>45 min 2 sec</td>
</tr>
<tr>
<td>Mission Restarts</td>
<td>0.180</td>
<td>0.127</td>
</tr>
</tbody>
</table>
Proposed CBR Wizard

- Will utilize a map-based interface and iconic task representations
- Will empower less-skilled robot commanders to develop sophisticated missions by avoiding the complexities of directly building FSAs
- Instead, users will build a mission by marking critical aspects on a map
- Case-Based Reasoner will fill in the gaps to form a complete mission plan.
CBR component of Wizard

- Relevant cases stored in standard relational database
- Two types of cases:
  - Task-based mission fragments (e.g., those from usability studies)
  - Location-based mission fragments (learned from experience in similar situations)
- Possible case indices:
  - type of mission (indoor, urban, wooded, etc.)
  - number of robots
  - localization requirements (accurate map, landmarks, etc.)
  - stealthiness of the mission
  - presence of enemy threats
- Case evaluation and adaptation currently being considered
Probabilistic Planning

• Partially Observable Markov Decision Processes (POMDPs) can model uncertainty for mobile robots
  – uncertainty due to sensor noise
  – actuator uncertainty
  – unpredictable events in the environment
• Hypothesis is that robots can act more robustly by modeling this uncertainty explicitly with POMDPs
• Distinctions between this and previous application of POMDPs to robot control:
  – Emphasis here has been on sensor-planning in the context of behavior-based robot systems
  – Solutions of POMDPs can be expressed as policy graphs, which are similar to the finite state automata used in MissionLab
Previously, we showed the creation of FSAs from policy graphs.

Recent efforts have included simulation runs and actual robot runs.

**POMDP Model**

Sensor model for scan:
\[ P(\text{detect-occupied} | \text{occupied}) = 0.8 \]
\[ P(\text{detect-empty} | \text{empty}) = 1.0 \]
Test scenario – cautious room entry

- Penalized heavily for entering an occupied room
- Equal chance of encountering an occupied or unoccupied room
- Observing room has small penalty and imperfect result (20% false negative)
Analysis & Simulation Results

- Analysis shows that multiple observations pay off as penalty for entering occupied room increases.
- Simulation study compared naïve "baseline" plan against POMDP-based "optimal" plan.
- Results correlated well with analysis.

<table>
<thead>
<tr>
<th>Reward for Entering Occupied Room x = -200</th>
<th>Baseline Plan</th>
<th>Optimal Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16.92</td>
<td>282.89</td>
</tr>
<tr>
<td>Reward for Entering Occupied Room x = -500</td>
<td>-1000.42</td>
<td>498.23</td>
</tr>
</tbody>
</table>
Actual Robot Results

- Used a Nomadic Technologies 150 equipped with a stereo microphone
- Robot proceeded down a hallway until it detected a door
- Robot stopped and listened for sound
  - For occupied rooms, a sound was generated every 10 seconds with 80% probability
- When executing the baseline plan, the robot would enter the room after one failure to detect noise
  - This caused the robot to incorrectly enter a room in 1 out of 5 attempts
- The POMDP-generated plan instructed the robot to sense 2-3 times
  - the robot correctly avoided occupied rooms in all trials
RL for Behavioral Assemblage Selection

• Essentially involves trial and error to determine when to switch from one behavior to another
• Operates at coarse granularity
  – implements behavioral assemblage selection
  – as opposed to parameterization, as is done in CBR/LM methods
• As reported previously:
  – Approach replaces the FSA with an interface allowing user to specify the environmental and behavioral states
  – Agent learns transitions between behavior states
  – Learning algorithm is implemented as an abstract module and different learning algorithms can be swapped in and out as desired.
RL test scenario

• An intelligent landmine
  – designed to intercept enemy tanks as they move down a nearby road and destroy them
  – idealized sensor determines the location of enemy tanks within a certain radius
  – sensor information is used with two perceptual triggers: CAN_INTERCEPT and NEAR

• Every timestep that the NEAR observation is made, the landmine receives a reward of +2

• The landmine is not penalized for any action.

• After making an observation and receiving a reward, the mine can choose WAIT or INTERCEPT
RL learning trials

• 750 learning scenarios
  – A learning scenario consists of 300 time steps in which the mine is attempting to intercept the tank

• Success of the Q-learner judged by the convergence properties of the Q-value table
Recent MARS-related publications


• Amin Atrash and Sven Koenig, “Probabilistic Planning for Behavior-Based Robotics,” to appear at the 14th International FLAIRS Conference.