Optimal Control of Non-deterministic Systems for a Fragment of Temporal Logic

Eric M. Wolff¹
Ufuk Topcu² and Richard M. Murray¹
¹Caltech and ²UPenn

CDC
11 December 2013
Modern Autonomous Systems

- How to specify **complex tasks**?
- How to create **optimal** controllers?
- React to **adversarial** environment?
UAV Surveillance Tasks

• Tracking a vehicle with a team of UAVs
Planning in a Dynamic Environment

• Dynamic obstacles + complex tasks in a warehouse

• Q: How to compute an optimal control policy that guarantees a complex, logical task is completed?
Our Contributions

• **Optimal** control for non-deterministic systems with *temporal logic* specifications
• **Polynomial** time controller synthesis
• **Anytime** optimization

<table>
<thead>
<tr>
<th>Cost function:</th>
<th>Average</th>
<th>Minimax</th>
<th>Task Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity:</td>
<td>POLY</td>
<td>POLY in task graph</td>
<td>EXP in task graph</td>
</tr>
</tbody>
</table>

Eric M. Wolff (Caltech)
Hierarchical Control Architecture

- We focus on the discrete planning layer
- Discrete plan is executed at continuous level

1. AlurHL00, BeltaH06, HabetsCS06, KaramanF09, KloetzerB08, WongpiromsarnTM12, and more

Eric M. Wolff (Caltech)
Non-deterministic Transition Systems

- A non-deterministic transition system (NTS) is a tuple $T = (S, A, R, s_0, AP, L)$ with
  - states $S$,
  - actions $A$,
  - transition function $R: S \times A \to 2^S$,
  - initial state $s_0$,
Non-deterministic Transition Systems

• A non-deterministic transition system (NTS) is a tuple $T = (S, A, R, s_0, \text{AP}, L)$ with
  – states $S$,
  – actions $A$,
  – transition function $R : S \times A \rightarrow 2^S$,
  – initial state $s_0$,
  – atomic propositions $\text{AP}$,
  – labeling function $L : S \rightarrow 2^{\text{AP}}$, and
  – non-negative valued cost function $c : S \times A \times S \rightarrow \mathbb{R}$.
Control Policies

- **Finite-memory control policy:** $\mu: S \times M \rightarrow A \times M$
- **Two-player game:**
  - **System** picks action using control policy
  - **Environment** picks next state
- $T^\mu(s)$: set of executions from state $s$ under policy $\mu$
Temporal Logic

- A logic for reasoning about how properties change over time
- Reason about infinite sequences $\sigma = s_0s_1s_2 \ldots$ of states
- Propositional logic: $\land$ (and), $\lor$ (or), $\Longrightarrow$ (implies), $\neg$ (not)
- Temporal operators: $U$ (until), $\circ$ (next), $\square$ (always), $\diamond$ (eventually)

Motion Planning

Dangerous liquid handling

Bomb disposal

Complex sequencing of actions
Temporal Logic

- A logic for reasoning about how properties change over time
- Reason about infinite sequences $\sigma = s_0 s_1 s_2 \ldots$ of states
- Propositional logic: $\land$ (and), $\lor$ (or), $\implies$ (implies), $\neg$ (not)
- Temporal operators: $\mathcal{U}$ (until), $\circ$ (next), $\square$ (always), $\lozenge$ (eventually)

Motion Planning

Dangerous liquid handling

Bomb disposal

INTRACTABLE!

Eric M. Wolff (Caltech)
Specification Language

- We consider formulas of the form:

\[ \varphi = \varphi_{\text{safe}} \land \varphi_{\text{resp}} \land \varphi_{\text{per}} \land \varphi_{\text{task}} \land \varphi_{\text{ss \ resp}}, \]

where

- \( \varphi_{\text{safe}} := \Box \psi_1 \), Safety
- \( \varphi_{\text{resp}} := \bigwedge_{j \in I_2} \Box (\psi_{2,j} \implies \Diamond \phi_{2,j}) \), Response
- \( \varphi_{\text{per}} := \Diamond \Box \psi_3 \), Stability
- \( \varphi_{\text{task}} := \bigwedge_{j \in I_4} \Box \Diamond \psi_{4,j} \), Repeated tasks
- \( \varphi_{\text{ss \ resp}} := \bigwedge_{j \in I_5} \Diamond \Box (\psi_{5,j} \implies \Diamond \phi_{5,j}) \) Steady-state response
Cost Functions

• Generic cost function $J$

\[ J : T^\mu (s) \rightarrow \mathbb{R}_{\geq 0} \]

• We consider:
  – Average cost
  – Minimax (bottleneck) cost
  – Average cost-per-task-cycle
Problem Statement

• **Given:**
  – Non-deterministic transition system $T$
  – Temporal logic specification $\varphi$ of the form
    \[
    \varphi = \varphi_{\text{safe}} \land \varphi_{\text{resp}} \land \varphi_{\text{per}} \land \varphi_{\text{task}} \land \varphi_{\text{resp}}^{\text{ss}}
    \]
  – Cost function $J$

• **Problem:** Create control policy $\mu$ such that the set of runs $T^{\mu}(s_0)$ satisfies $\varphi$ and minimizes $J$

\[
\min_{\mu} J(T^{\mu}(s_0)) \\
\text{s.t.} \quad T^{\mu}(s_0) \models \varphi
\]
Related Work

• Automata-based approach [Vardi & Wolper]
  – Construct automaton from specification
  – EXP or 2-EXP in the specification

• Our approach
  – No automaton construction
  – Compute directly on the state space
Related Work

• Related logics:
  – GR(1): PitermanPS06, BloemJPPS12
  – GRabin(1): Ehlers11
  – AlurT04; MalerPS95

• Optimal control: JingEKG13

• How this work differs:
  – More system properties/tasks than GR(1)
  – Only bounded liveness assumptions on environment
Main Idea

• Optimization boils down to reasoning about worst-case costs between tasks
• Use value function and task graph for this

\[ \varphi_{\text{task}} := \bigwedge_{j \in I_4} \Box \Diamond \psi_{4,j} \]

Tasks: P, D0, D1, D2, D3
Value Function and Reachability

• $V^c_B(s)$: minimum cost to reach set B from state s under all resolutions of the non-determinism

$$V^c_{B,T}(s) = \min_{a \in A(s)} \max_{t \in R(s,a)} V^c_{B,T}(t) + c(s,a,t)$$
Value Function and Reachability

- \( V^c_B(s) \): minimum cost to reach set B from state s under all resolutions of the non-determinism

\[
V^c_B,\mathcal{T}(s) = \min_{a \in A(s)} \max_{t \in R(s,a)} V^c_B,\mathcal{T}(t) + c(s,a,t)
\]

- Example
  - \( V^c_4(1) = \infty \)
  - \( V^c_4(2) = \infty \)
  - \( V^c_4(3) = 1 \)
  - \( V^c_4(4) = 0 \)
Task Graph

• Create new graph that encodes shortest paths between tasks

Tasks: A,B
Cost = 2

Eric M. Wolff (Caltech)
Task Graph

- Create new graph that encodes shortest paths between tasks

Tasks: A, B
Cost = 1.5

Including subsets in the task graph can reduce cost!

Eric M. Wolff (Caltech)
Task Graph

- Create new graph that encodes shortest paths between tasks
- Number of states
  - Deterministic: |F|
  - Non-deterministic: \( \sum_{i \in I_4} 2^{|F_i|} - 1 \)

\[
\varphi_{\text{task}} := \bigwedge_{j \in I_4} \Box \Diamond \psi_{4,j}.
\]
Average Cost Function

• Average cost of run $\sigma$ is

$$J'_{\text{avg}}(\sigma, \mu(\sigma)) := \lim_{n \to \infty} \sup \frac{\sum_{t=0}^{n} c(\sigma_t, \mu(\sigma_t), \sigma_{t+1})}{n}$$

• The average cost function is

$$J_{\text{avg}}(\mathcal{T}^\mu(s)) := \sup_{\sigma \in \mathcal{T}^\mu(s_0)} J'_{\text{avg}}(\sigma, \mu(\sigma))$$

Eric M. Wolff (Caltech)
Average—Solution

- Policy has two parts:
  1) Optimal policy ignoring the tasks
  2) Visit all tasks once
- An optimal policy alternates
  - $121121112...$
  - Requires infinite memory
- We adapt an algorithm from Chatterjee, Henzinger, Jurdzinski 2006.
- Polynomial time
Minimax (bottleneck) Cost

• Minimax cost of run $\sigma$ is

$$J'_\text{bot}(\sigma, \mu(\sigma)) := \limsup_{i \to \infty} (T_{\text{task}}(i + 1) - T_{\text{task}}(i))$$

where $T_{\text{task}}(i)$ is the accumulated cost at the i-th completion of a task

• The minimax cost function is

$$J_{\text{bot}}(\mathcal{T}^\mu(s)) := \max_{\sigma \in \mathcal{T}^\mu(s)} J'_\text{bot}(\sigma, \mu(\sigma))$$

Eric M. Wolff (Caltech)
Minimax—Solution

• **Approach**
  – Fix a cost $\lambda$
  – Remove all edges with cost $>\lambda$ from task graph
  – Is remaining graph have a strongly connected component that includes all tasks?
  – Binary search on $\lambda$

• **Polynomial time** in task graph

---

Eric M. Wolff (Caltech)
Average Cost-Per-Task-Cycle

- Average cost-per-task-cycle of run $\sigma$ is

$$J_{TC}'(\sigma, \mu(\sigma)) := \limsup_{n \to \infty} \frac{\sum_{t=0}^{n} c(\sigma_t, \mu(\sigma_t), \sigma_{t+1})}{\sum_{t=0}^{n} I_{TC}(t)}$$

- The average cost-per-task-cycle cost function is

$$J_{TC}(\mathcal{T}^\mu(s)) := \max_{\sigma \in \mathcal{T}^\mu(s)} J_{TC}'(\sigma, \mu(\sigma))$$
Optimality for Task Cycle is Hard

• **Theorem:** Computing a control policy that minimizes the average cost-per-task-cycle is NP-hard, even in the deterministic case.

• **Proof:** Construct a generalized traveling salesman problem where tasks are nodes in the TSP graph.

[Diagram of a generalized TSP problem]
Task Cycle—Solution

• **Assumption**: The task ordering is fixed

• Solve generalized TSP on task graph
  – Use commercial solvers
  – Approximate solutions

• Solution gives optimal task ordering
Example: Pickup and Delivery

• System:
  – Robot and obstacle move to adjacent regions each step

• Specs:
  – Always avoid collisions
  – Repeatedly visit Pickup and Dropoff locations

Eric M. Wolff (Caltech)
Example: Pickup and Delivery

- **System:**
  - Robot and obstacle move to adjacent regions each step

- **Specs:**
  - Avoid collisions
  - Repeatedly visit Pickup and Dropoff locations

![Computation Time—Optimal Controller](image)

Eric M. Wolff (Caltech)
Comparison to GR1 (feasible)

![Graph comparing time to grid size and number of states between previous work and our work. The graph shows that our work is more efficient, especially as the number of states increases.]
Conclusions

• Optimal control with temporal logic

<table>
<thead>
<tr>
<th>Cost function:</th>
<th>Average</th>
<th>Minimax</th>
<th>Task Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity:</td>
<td>POLY</td>
<td>POLY in task graph</td>
<td>EXP in task graph</td>
</tr>
</tbody>
</table>

• Future work
  – Receding horizon control
  – Removing fixed-ordering assumption

Eric M. Wolff (Caltech)
Thank you!

• **Contact:** Eric M. Wolff
  - Email: [ewolff@caltech.edu](mailto:ewolff@caltech.edu)
  - Web: [www.cds.caltech.edu/~ewolff/](http://www.cds.caltech.edu/~ewolff/)

• **Funding:** NDSEG fellowship, Boeing, AFOSR