Motion Planning for LTL Specifications: A Satisfiability Modulo Convex Optimization Approach

Yasser Shoukry
UC Berkeley, UCLA, and UPenn

Joint work with

Pierluigi Nuzzo (UC Berkeley), Indranil Saha (IIT Kanpur), Alberto Sangiovanni-Vincentelli (UC Berkeley), Sanjit A. Seshia (UC Berkeley), George Pappas (UPenn), and Paulo Tabuada (UCLA)
Motion Planning Problem

Given:

Robot Dynamics (with input constraints):
\[ x_{t+1} = Ax_t + Bu_t, \]
\[ x_0 = x, \]
\[ \|u_t\|_\infty \leq u \forall t \in \mathbb{N} \]

Workspace \( W \):
all obstacles are assumed to be unions of polyhedra.

Atomic propositions \( \Pi = \{ \pi_1, ..., \pi_m \} \): defined over the free-workspace.

LTL Specification \( \Phi \):
for simplicity, I will focus on reach-avoid problems:
\[ \Diamond \pi_1 \land \Box \neg \pi_0. \]

Objective:
Generate the input sequence \( u_0, u_1, ..., u_L \) such that the trajectory of the robot satisfies \( \Phi \).
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):
  \[ x_{t+1} = Ax_t + Bu_t, \quad x_0 = \bar{x}, \quad \|u_t\|_\infty \leq \bar{u} \quad \forall t \in \mathbb{N} \]
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):
  \[ x_{t+1} = Ax_t + Bu_t, \quad x_0 = \bar{x}, \quad \|u_t\|_\infty \leq \bar{u} \quad \forall t \in \mathbb{N} \]

- Workspace \( \mathcal{W} \):

  ![Workspace Diagram]

Objective:

Generate the input sequence \( u_0, u_1, \ldots, u_L \) such that the trajectory of the robot satisfies \( \Phi \).
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):
  \[ x_{t+1} = Ax_t + Bu_t, \quad x_0 = \bar{x}, \quad \|u_t\|_\infty \leq \bar{u} \quad \forall t \in \mathbb{N} \]

- Workspace \( \mathcal{W} \): all obstacles are assumed to be unions of polyhedra.
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):
  \[ x_{t+1} = Ax_t + Bu_t, \quad x_0 = \bar{x}, \quad \|u_t\|_{\infty} \leq \bar{u} \quad \forall t \in \mathbb{N} \]

- Workspace \( \mathcal{W} \): all obstacles are assumed to be unions of polyhedra.

- Atomic propositions \( \Pi = \{\pi_1, \ldots, \pi_m\} \): defined over the free-workspace.
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):
  \[ x_{t+1} = Ax_t + Bu_t, \quad x_0 = \bar{x}, \quad \|u_t\|_\infty \leq \bar{u} \quad \forall t \in \mathbb{N} \]

- Workspace \( \mathcal{W} \): all obstacles are assumed to be unions of polyhedra.

- Atomic propositions \( \Pi = \{\pi_1, \ldots, \pi_m\} \): defined over the free-workspace.

- LTL Specification \( \Phi \):

  \[ \Box \pi_1 \land \neg \Box \pi_0. \]
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):
  \[ x_{t+1} = Ax_t + Bu_t, \quad x_0 = \bar{x}, \quad \|u_t\|_\infty \leq \bar{u} \quad \forall t \in \mathbb{N} \]
  
- Workspace \( \mathcal{W} \): all obstacles are assumed to be unions of polyhedra.

- Atomic propositions \( \Pi = \{\pi_1, \ldots, \pi_m\} \): defined over the free-workspace.

- LTL Specification \( \Phi \):
  - for simplicity, I will focus on reach-avoid problems: \( \Diamond \pi_1 \land \Box \neg \pi_0 \).
Motion Planning Problem

**Given:**
- **Robot Dynamics (with input constraints):**
  \[ x_{t+1} = Ax_t + Bu_t, \quad x_0 = \bar{x}, \quad \|u_t\|_\infty \leq \bar{u} \quad \forall t \in \mathbb{N} \]
- **Workspace \( \mathcal{W} \):** all obstacles are assumed to be unions of polyhedra.
- **Atomic propositions \( \Pi = \{\pi_1, \ldots, \pi_m\} \):** defined over the free-workspace.
- **LTL Specification \( \Phi \):**
  - for simplicity, I will focus on reach-avoid problems: \( \Diamond \pi_1 \land \Box \neg \pi_0 \).

**Objective:**
Motion Planning Problem

Given:
- Robot Dynamics (with input constraints):
  \[ x_{t+1} = Ax_t + Bu_t, \quad x_0 = \bar{x}, \quad \|u_t\|_\infty \leq \bar{u} \quad \forall t \in \mathbb{N} \]
- Workspace \( \mathcal{W} \): all obstacles are assumed to be unions of polyhedra.
- Atomic propositions \( \Pi = \{\pi_1, \ldots, \pi_m\} \): defined over the free-workspace.
- LTL Specification \( \Phi \):
  - for simplicity, I will focus on reach-avoid problems: \( \Diamond \pi_1 \land \Box \neg \pi_0 \).

Objective:
- Generate the input sequence \( u_0, u_1, \ldots, u_L \) such that the trajectory of the robot satisfies \( \Phi \).
Solver Architecture [0/3]

- **Step 0**: Compute coarse, obstacle-based, discretization of the free space.
**Step 0:** Compute coarse, obstacle-based, discretization of the free space.
 Solver Architecture [0/3]

- **Step 0:** Compute coarse, obstacle-based, discretization of the free space.
- Avoid discrete state-space explosion (no abstraction of continuous dynamics, no grid-based abstractions).
Step 0: Compute coarse, obstacle-based, discretization of the free space.

Avoid discrete state-space explosion (no abstraction of continuous dynamics, no grid-based abstractions).
**Step 1:** For a given horizon $L$, use a SAT solver to find a candidate high-level path (Bounded Model Checking) that satisfies the LTL specification.
Solver Architecture [1/3]

- **Step 1:** For a given horizon $L$, use a SAT solver to find a candidate high-level path (Bounded Model Checking) that satisfies the LTL specification.
  - Increase the horizon $L$ until such high-level path is found.
Step 1: For a given horizon $L$, use a SAT solver to find a candidate high-level path (Bounded Model Checking) that satisfies the LTL specification.

- Increase the horizon $L$ until such high-level path is found.
Step 2: Check whether the high-level trajectory $\rho$ is feasible (satisfies the robot initial state, dynamics, and input constraints).
Step 2: Check whether the high-level trajectory \( \rho \) is feasible (satisfies the robot initial state, dynamics, and input constraints).

Can be casted as an optimization problem.

**Problem (\texttt{CON-PLAN.CHECK})**

\[
\begin{align*}
\min & \quad 1 \\
\text{subject to} & \quad i = 0, \ldots, L \\
\text{(initial condition)} & \quad x_0 = \bar{x}, \\
\text{(dynamics constraints)} & \quad x_{i+1} = Ax_i + Bu_i \\
\text{(input constraints)} & \quad \|u_i\| \leq \bar{u}, \\
\text{(plan constraints)} & \quad x_i \in \rho_i
\end{align*}
\]
Step 2: Check whether the high-level trajectory $\rho$ is feasible (satisfies the robot initial state, dynamics, and input constraints).

- Can be casted as a convex optimization problem (Linear Program).

Problem (CON-PLAN.CHECK)

$$\min \quad 1$$

(initial condition) $x_0 = \bar{x}$,

(dynamics constraints) $x_{i+1} = Ax_i + Bu_i$

(input constraints) $\|u_i\| \leq \bar{u}$,

(plan constraints) $x_i \in \rho_i$

$\rho = \pi_1, \pi_17, \ldots, \pi_{28}$

$u_0, u_1, \ldots, u_{L-1}$

$\mathbf{A}, \mathbf{B}, \bar{x}, \bar{u}$
**Step 3:** Generate counter example.

\[ \phi_{\text{triv-ce}} := \bigvee_{i=0}^{L} \neg \rho_i, \]

\[ \rho = \pi_1, \pi_{17}, \ldots, \pi_{28} \]
Step 3: Generate counter example.

\[ \phi_{\text{triv-ce}} := \bigvee_{i=0}^{L} \neg \rho_i, \]

Repeat.
Step 3: Generate counter example.

\[ \phi_{\text{triv-ce}} := \bigvee_{i=0}^{L} \neg \rho_i, \]

Repeat.
Detecting the *earliest possible* occurrence of an infeasible transition between two regions rules out a *broader class* of assignments to Boolean variables.
Detecting the earliest possible occurrence of an infeasible transition between two regions rules out a broader class of assignments to Boolean variables.

Example 1: \( \phi_{\text{counter-example}} := \neg \rho_0 \lor \neg \rho_1 \lor \neg \rho_2 \lor \neg \rho_3 \)
Detecting the earliest possible occurrence of an infeasible transition between two regions rules out a broader class of assignments to Boolean variables.

Example 1: $\phi_{\text{counter-example}} := \neg \rho_0 \lor \neg \rho_1 \lor \neg \rho_2 \lor \neg \rho_3$

Example 2: $\phi_{\text{counter-example}} := \neg \rho_0 \lor \neg \rho_1$
Searching for succinct counterexample can be performed by checking the feasibility of prefixes of \( \rho \).

Leads to solving multiple optimization problems.
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of $\rho$.
- Leads to solving multiple optimization problems.

\[ \cdots \]

\[ \cdots \]
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of $\rho$.
- Leads to solving multiple optimization problems.
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of $\rho$.
- Leads to solving multiple optimization problems.
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of $\rho$.
- Leads to solving multiple optimization problems.
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of ρ.
- Leads to solving multiple optimization problems.
Searching for succinct counterexample can be performed by checking the feasibility of prefixes of $\rho$.

Leads to solving multiple optimization problems.
**Key insight:** we can **check feasibility** of high-level plans and generate the **shortest counter example** by solving a single linear program.

\[
\begin{align*}
\text{min} & \quad u_0, \ldots, u_L \in \mathbb{R}^m \\
& \quad v_0, \ldots, v_L \in \mathbb{R}^m \\
& \quad s_0^u, \ldots, s_L^u \in \mathbb{R}^m \\
& \quad s_0^v, \ldots, s_L^v \in \mathbb{R}^m \\
& \quad x_0, \ldots, x_{L+1} \in \mathbb{R}^n \\
\text{subject to} & \quad x_0 = \vec{x}, \\
& \quad h_{x \rightarrow \gamma W}(x_i) \in \rho_i^W, \quad i = 1, \ldots, L + 1 \\
& \quad x_{i+1} = Ax_i + Bu_i + B'v_i \quad i = 0, \ldots, L \\
& \quad \|u_i\| \leq \bar{u} + s_i^u, \quad i = 0, \ldots, L \\
& \quad \|v_i\| \leq s_i^v, \quad i = 0, \ldots, L \\
& \quad 0 \leq s_i^u, \quad 0 \leq s_i^v, \quad i = 0, \ldots, L \\
& \quad \frac{\varepsilon}{\varepsilon} \left( \sum_{k=0}^{L} s_k^u + s_k^v \right) \leq s_i^u + s_i^v \quad i = 1, \ldots, L
\end{align*}
\]
**Key insight:** we can check feasibility of high-level plans and generate the shortest counter example by solving a single linear program.

Satisfiability modulo convex-optimization approach.

\[
\begin{align*}
\min & \quad v_0, \ldots, v_L \in \mathbb{R}^m, \\
& \quad s_0^\mu, \ldots, s_L^\mu \in \mathbb{R} \\
& \quad s_0^\nu, \ldots, s_L^\nu \in \mathbb{R} \\
& \quad x_0, \ldots, x_L+1 \in \mathbb{R}^n
\end{align*}
\]

subject to

\[
\begin{align*}
& x_0 = \bar{x}, \\
& h_{x \rightarrow \gamma}(x_i) \in \rho_i^W, \quad i = 1, \ldots, L + 1 \\
& x_{i+1} = Ax_i + Bu_i + B'v_i \quad i = 0, \ldots, L \\
& ||u_i|| \leq \bar{u} + s_i^\mu, \quad i = 0, \ldots, L \\
& ||v_i|| \leq s_i^\nu, \quad i = 0, \ldots, L \\
& 0 \leq s_i^\mu, \quad 0 \leq s_i^\nu \quad i = 0, \ldots, L \\
& \frac{\epsilon}{\epsilon} \left( \sum_{k=0}^{L-1} s_k^\mu + s_k^\nu \right) \leq s_i^\mu + s_i^\nu \quad i = 1, \ldots, L
\end{align*}
\]
Case Study 1: Dubin’s Vehicle

- **Robot dynamics** (Dubin car):
  \[
  \dot{x} = v \cos \theta \quad \dot{y} = v \sin \theta \quad \dot{\theta} = \omega
  \]

  Dynamics can be transformed into a linear chain of integrators using dynamic feedback linearization.

Workspace: 30m × 30m maze-like workspace.

We increase number of passages from 1 to 4.

We compare execution-time against (1) standard RRT and (2) LTL OPT tool (mixed-integer linear program).
Case Study 1: Dubin’s Vehicle

- **Robot dynamics** (Dubin car): $\dot{x} = v \cos \theta$, $\dot{y} = v \sin \theta$, $\dot{\theta} = \omega$

- Dynamics can be transformed into a linear chain of integrators using dynamic feedback linearization.
Case Study 1: Dubin’s Vehicle

- **Robot dynamics** (Dubin car): \( \dot{x} = v \cos \theta \quad \dot{y} = v \sin \theta \quad \dot{\theta} = \omega \)

  Dynamics can be transformed into a linear chain of integrators using dynamic feedback linearization.

- **Workspace**: 30m × 30m maze-like workspace.

  We increase number of passages from 1 to 4.
Case Study 1: Dubin’s Vehicle

- **Robot dynamics** (Dubin car): \[ \dot{x} = v \cos \theta \quad \dot{y} = v \sin \theta \quad \dot{\theta} = \omega \]
  - Dynamics can be transformed into a linear chain of integrators using dynamic feedback linearization.

- **Workspace**: 30m × 30m maze-like workspace.
  - We increase number of passages from 1 to 4.
  - We compare execution-time against (1) standard RRT and (2) LTL OPT tool (mixed-integer linear program).

<table>
<thead>
<tr>
<th>Number of passages</th>
<th>SMT-Based Motion Planner [s]</th>
<th>RRT [s]</th>
<th>LTL OPT [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Discrete abstraction</td>
<td>DIS-PLAN</td>
<td>CON-PLAN</td>
</tr>
<tr>
<td>1</td>
<td>1.9975</td>
<td>0.1360</td>
<td>0.2542</td>
</tr>
<tr>
<td>2</td>
<td>7.1461</td>
<td>1.1290</td>
<td>0.9294</td>
</tr>
<tr>
<td>3</td>
<td>19.3267</td>
<td>3.6495</td>
<td>1.0053</td>
</tr>
<tr>
<td>4</td>
<td>43.0985</td>
<td>4.0913</td>
<td>1.9204</td>
</tr>
</tbody>
</table>
Case Study 2: Scalability Results

- Maze-like workspace with increasing number of passages.
- We increase the number of states $n$ (randomly generate the matrices $A$ and $B$).
- Take average across 10 runs.

![Graph showing execution time vs. number of states for different maze configurations.](image)
Final Remarks

- What is still missing?
Final Remarks

What is still missing?

- We need to compare with other methods like RRT* and PRM.
Final Remarks

What is still missing?

- We need to compare with other methods like RRT* and PRM.
- We need to compare against existing benchmarks.
Final Remarks

- **What is still missing?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.

- SAT + Convex optimization = tools:
Final Remarks

What is still missing?
- We need to compare with other methods like RRT* and PRM.
- We need to compare against existing benchmarks.

SAT + Convex optimization \( \equiv \) tools:
- Probabilistic CTL (PCTL) verification of Markov Decision Processes.
Final Remarks

What is still missing?
- We need to compare with other methods like RRT* and PRM.
- We need to compare against existing benchmarks.

SAT + Convex optimization = tools:
- Probabilistic CTL (PCTL) verification of Markov Decision Processes.
- Security: secure state estimation.
Final Remarks

- **What is still missing?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.

- **SAT + Convex optimization**

  - Probabilistic CTL (PCTL) verification of Markov Decision Processes.
  - Security: secure state estimation.
  - Sensor Networks: localization.
Final Remarks

- **What is still missing?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.

- **SAT + Convex optimization ? tools:**
  - Probabilistic CTL (PCTL) verification of Markov Decision Processes.
  - Security: secure state estimation.
  - Sensor Networks: localization.
  - Motion planning (this work).
Final Remarks

- **What is still missing?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.

- **SAT + Convex optimization = tools:**
  - Probabilistic CTL (PCTL) verification of Markov Decision Processes.
  - Security: secure state estimation.
  - Sensor Networks: localization.
  - Motion planning (this work).
  - We are currently developing a comprehensive theory of Satisfiability Modulo Convex Optimization.