Review Papers on HJ Reachability

• Bansal, Chen, Herbert, Tomlin. “Hamilton-Jacobi reachability: A brief overview and recent advances,” 2017

• Chen, Tomlin. “Hamilton-Jacobi Reachability: Some Recent Theoretical Advances and Applications in Unmanned Airspace Management,” 2018
Terminology

• Minimal backward reachable set
  • $\mathcal{A}(t) = \{\bar{x}: \exists \Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x, u, d), x(t) = \bar{x}, x(0) \in T\}$
  • Control minimizes size of reachable set

• Maximal backward reachable set
  • $\mathcal{R}(t) = \{\bar{x}: \forall \Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x, u, d), x(t) = \bar{x}, x(0) \in T\}$
  • Control maximizes size of reachable set
### Reaching vs. Avoiding

- **Avoiding danger**
  - BRS definition
    \[ \mathcal{A}(t) = \{ \bar{x} : \exists \Gamma[u] \cdot \forall u \cdot \bar{x} = f(x, u, d), x(t) = \bar{x}, x(0) \in \mathcal{T} \} \]
  - Value function
    \[ V(t, x) = \min_{\Gamma[u]} \max_u l(x(0)) \]
  - HJ PDE
    \[ \frac{\partial V}{\partial t} + \max_u \min_d \left[ (\frac{\partial V}{\partial x})^\top f(x, u, d) \right] = 0 \]
  - Optimal control
    \[ u^* = \arg \max_u \min_d \left( \frac{\partial V}{\partial x} \right)^\top f(x, u, d) \]

- **Reaching a goal**
  - BRS definition
    \[ \mathcal{R}(t) = \{ \bar{x} : \forall \Gamma[u] \cdot \exists u \cdot \bar{x} = f(x, u, d), x(t) = \bar{x}, x(0) \in \mathcal{T} \} \]
  - Value function
    \[ V(t, x) = \max_{\Gamma[u]} \min_u l(x(0)) \]
  - HJ PDE
    \[ \frac{\partial V}{\partial t} + \min_u \max_d \left[ (\frac{\partial V}{\partial x})^\top f(x, u, d) \right] = 0 \]
  - Optimal control
    \[ u^* = \arg \min_u \max_d \left( \frac{\partial V}{\partial x} \right)^\top f(x, u, d) \]
“Sets” vs. “Tubes”

• Backward reachable set (BRS)
  • Only final time matters
  • Initial states that passing through target are not necessarily in BRS
  • Not ideal for safety

\[
\begin{align*}
D_0 & \quad \text{target set} \\
E_0 & \quad \text{target set} \\
l(x) & \leq 0, \\
V(x_g(0), 0) & > 0 \\
V(x_b(0), 0) & \leq 0 \\
V(x_B(0), 0) & > 0
\end{align*}
\]

• Backward reachable tube (BRT)
  • Keep track of entire time duration
  • Initial states that pass thorough target are in BRT
  • Used to make safety guarantees

\[
\begin{align*}
D_0 & \quad \text{target set} \\
E_0 & \quad \text{target set} \\
l(x) & \leq 0, \\
V(x_g(0), 0) & > 0 \\
V(x_b(0), 0) & \leq 0 \\
V(x_B(0), 0) & \leq 0
\end{align*}
\]
Terminology

• Minimal backward reachable set
  - $\mathcal{A}(t) = \{\bar{x}: \exists \Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x, u, d), x(t) = \bar{x}, x(0) \in \mathcal{T}\}$
  - Control minimizes size of reachable set

• Maximal backward reachable set
  - $\mathcal{R}(t) = \{\bar{x}: \forall \Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x, u, d), x(t) = \bar{x}, x(0) \in \mathcal{T}\}$
  - Control maximizes size of reachable set

• Minimal and maximal backward reachable tube
  - $\bar{\mathcal{A}}(t) = \{\bar{x}: \exists \Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x, u, d), x(t) = \bar{x}, \exists s \in [t, 0], x(s) \in \mathcal{T}\}$
  - $\bar{\mathcal{R}}(t) = \{\bar{x}: \forall \Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x, u, d), x(t) = \bar{x}, \exists s \in [t, 0], x(s) \in \mathcal{T}\}$
“Sets” vs. “Tubes”

- Backward reachable set (BRS)
  - Value function definition
    - \( V(t, x) = \min_{\Gamma[u]} \max_{u} \max_{d} l(x(0)) \)
  - Value function obtained from
    \[
    \frac{\partial V}{\partial t} + \max_{u} \min_{d} \left[ \left( \frac{\partial V}{\partial x} \right)^{\top} f(x, u, d) \right] = 0
    \]

- Backward reachable tube (BRT)
  - Value function definition
    - \( V(t, x) = \min_{\Gamma[u]} \max_{u} \min_{s \in [t, 0]} l(x(s)) \)
  - Value function obtained from
    \[
    \min_{\Gamma[u]} \left\{ \frac{\partial V}{\partial t} + \max_{u} \min_{d} \left[ \left( \frac{\partial V}{\partial x} \right)^{\top} f(x, u, d) \right], l(x) - V(t, x) \right\} = 0
    \]
Reaching vs. Avoiding: Backward Reachable Tubes

- Avoiding danger
  
  BRT definition
  \[ \mathcal{A}(t) = \left\{ \dot{x} : \exists \Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x, u, d), x(t) = \bar{x}, \exists s \in [t, 0], x(s) \in T \right\} \]
  
  Value function
  \[ V(t, x) = \min_{\Gamma[u]} \max_u \min_{s \in [t, 0]} l(x(s)) \]
  
  HJ Variational Inequality
  \[ \min_u \left\{ \frac{\partial V}{\partial t} + \max_d \left[ \left( \frac{\partial V}{\partial x} \right)^{\top} f(x, u, d) \right], l(x) - V(t, x) \right\} = 0 \]
  
  Optimal control
  \[ u^* = \arg \max_u \min_d \left( \frac{\partial V}{\partial x} \right)^{\top} f(x, u, d) \]

- Reaching a goal
  
  BRT definition
  \[ \mathcal{R}(t) = \left\{ \dot{x} : \forall \Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x, u, d), x(t) = \bar{x}, \exists s \in [t, 0], x(s) \in T \right\} \]
  
  Value function
  \[ V(t, x) = \max_{\Gamma[u]} \min_u \min_{s \in [t, 0]} l(x(s)) \]
  
  HJ Variational Inequality
  \[ \min_u \left\{ \frac{\partial V}{\partial t} + \min_d \max_{\Gamma[u]} \left[ \left( \frac{\partial V}{\partial x} \right)^{\top} f(x, u, d) \right], l(x) - V(t, x) \right\} = 0 \]
  
  Optimal control
  \[ u^* = \arg \min_u \max_d \left( \frac{\partial V}{\partial x} \right)^{\top} f(x, u, d) \]
Hamilton-Jacobi (HJ) Reachability Theory

Target set: $\mathcal{T} = \{x: l(x) < 0\}$

Value function: $V(t, x)$

$$V(t, x(t)) = \min_{\Gamma[u]} \max_{u(\cdot)} \min_{s \in [t, 0]} l(x(s))$$

subject to $\dot{x} = f(x, u, d), t \leq 0$

$x_1(t) \in \mathcal{R}(t) \iff V(t, x_1(t)) < 0$

$x_2(t) \notin \mathcal{R}(t) \iff V(t, x_2(t)) \geq 0$
Hamilton-Jacobi (HJ) Reachability Theory

- Hamilton-Jacobi Variational Inequality:
  \[
  \min \left\{ \frac{\partial V}{\partial t} + \max_u \min_d \left[ \left( \frac{\partial V}{\partial x} \right)^T f(x, u, d) \right], l(x) - V(t, x) \right\} = 0, \quad t \leq 0
  \]
  \[
  V(0, x) = l(x)
  \]

Target set: \( \mathcal{T} = \{x: l(x) < 0\} \)

Minimal backward reachable tube: \( \mathcal{R}(t) = \{x: V(t, x) < 0\} \)
Hamilton-Jacobi (HJ) Reachability Theory

- Hamilton-Jacobi Variational Inequality:
  \[
  \min \left\{ \frac{\partial V}{\partial t} + \max_{u, d} \min_{x} \left[ \left( \frac{\partial V}{\partial x} \right)^T f(x, u, d) \right], l(x) - V(t, x) \right\} = 0, \quad t \leq 0
  \]
  \[
  V(0, x) = l(x)
  \]

  Target set:
  \[
  \mathcal{T} = \{ x : l(x) < 0 \}
  \]

  Minimal backward reachable tube:
  \[
  \mathcal{R}(t) = \{ x : V(t, x) < 0 \}
  \]
Hamilton-Jacobi (HJ) Reachability Theory

- Hamilton-Jacobi Variational Inequality:
\[
\min \left\{ \frac{\partial V}{\partial t} + \max_u \min_d \left[ \left( \frac{\partial V}{\partial x} \right)^T f(x, u, d) \right], l(x) - V(t, x) \right\} = 0, \quad t \leq 0
\]

\[
V(0, x) = l(x)
\]

Minimal backward reachable tube:
\[
\mathcal{R}(t) = \{ x : V(t, x) < 0 \}
\]
Hamilton-Jacobi (HJ) Reachability Theory

- Hamilton-Jacobi Variational Inequality:
\[
\min \left\{ \frac{\partial V}{\partial t} + \max_u \min_d \left[ \left( \frac{\partial V}{\partial x} \right)^\top f(x, u, d) \right], l(x) - V(t, x) \right\} = 0, \quad t \leq 0
\]
\[
V(0, x) = l(x)
\]

Minimal backward reachable tube:
\[
\mathcal{R}(t) = \{ x : V(t, x) < 0 \} 
\]
Hamilton-Jacobi (HJ) Reachability Theory

- Hamilton-Jacobi Variational Inequality:
  \[
  \min \left\{ \frac{\partial V}{\partial t} + \max_u \min_d \left[ \left( \frac{\partial V}{\partial x} \right)^\top f(x, u, d) \right] , l(x) - V(t, x) \right\} = 0, \ t \leq 0
  \]
  \[
  V(0, x) = l(x)
  \]
  Minimal backward reachable tube:
  \[
  R(t) = \{ x : V(t, x) < 0 \} 
  \]
Intruder Avoidance

Platoon leader (autonomous)

Intruder (human controlled)
“Flavours” of Reachability

So far:

<table>
<thead>
<tr>
<th></th>
<th>Backward reachable set</th>
<th>Backward reachable tube</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal</td>
<td>Avoid $\mathcal{A}(t)$ to stay safe at $t = 0$</td>
<td>Avoid $\overline{\mathcal{A}}(t)$ to stay safe during $[t, 0]$</td>
</tr>
<tr>
<td>Maximal</td>
<td>Be in $\mathcal{R}(t)$ to reach goal at $t = 0$</td>
<td>Be in $\overline{\mathcal{R}}(t)$ to reach goal during $[t, 0]$</td>
</tr>
</tbody>
</table>

Other variations:

- Forward reachable sets and tubes
- Reach-avoid sets and tubes
  - States from which goal can be reached while avoiding obstacles
Wind speed: 11 m/s

6 m/s

Scheduled arrival time separation: 0 s
5 s
10 s
Comments

• Computational complexity
  • $V(t, x)$ is computed on an $(n + 1)$-dimensional grid
  • Currently, $n \leq 5$ is possible. GPU acceleration under-way
  • Dimensionality reduction methods sometimes help

• Related approaches
  • Sacrifice global optimality
  • Give up guarantees
  • Sampling-based methods
  • Reinforcement learning

\[ O(N^d) \] time and space complexity!

- **1D**: $< 0.1s$ negligible RAM
- **2D**: seconds tens of megabytes
- **3D**: minutes hundreds of megabytes
- **4D**: hours hundreds of megabytes
- **5D**: days gigabytes
- **6D**: intractable!
Numerical Toolboxes

• helperOC Matlab toolbox
  • https://github.com/HJReachability/helperOC.git
  • Reachability wrapper around the level set toolbox
  • Requires level set toolbox
    • Hamilton-Jacobi PDE solver by Ian Mitchell, UBC
    • https://bitbucket.org/ian_mitchell/toolboxls

• C++, CUDA, Julia version in development, beta available for C++
  • C++: 5+ times faster than Matlab
  • CUDA: Up to 100 times faster than Matlab
  • https://github.com/HJReachability/beacls
function tutorial()
% 1. Run Backward Reachable Set (BRS) with a goal
  uMode = 'min' <-- goal
  minWith = 'none' <-- Set (not tube)
  compTraj = false <-- no trajectory
% 2. Run BRS with goal, then optimal trajectory
  uMode = 'min' <-- goal
  minWith = 'none' <-- Set (not tube)
  compTraj = true <-- compute optimal trajectory
% 3. Run Backward Reachable Tube (BRT) with a goal, then optimal trajectory
  uMode = 'min' <-- goal
  minWith = 'minWithTarget' <-- Tube (not set)
  compTraj = true <-- compute optimal trajectory
% 4. Add disturbance
  dStep1: define a dMax (dMax = [.25, .25, 0];)
  dStep2: define a dMode (opposite of uMode)
  dStep3: input dMax when creating your DubinsCar
  dStep4: add dMode to schemeData
% 5. Change to an avoid BRT rather than a goal BRT
  uMode = 'max' <-- avoid
  dMode = 'min' <-- opposite of uMode
  minWith = 'minWithTarget' <-- Tube (not set)
  compTraj = false <-- no trajectory
% 6. Change to a Forward Reachable Tube (FRT)
  add schemeData.tMode = 'forward'
  note: now having uMode = 'max' essentially says "see how far I can reach"
% 7. Add obstacles
  add the following code:
  obstacles = shapeCylinder(g, 3, [-1.5; 1.5; 0], 0.75);
  HJExtraArgs.obstacles = obstacles;
% 8. Add random disturbance (white noise)
  add the following code:
  HJExtraArgs.addGaussianNoiseStandardDeviation = [0; 0; 0.5];
Tutorial Code Overview

Computation domain
- Make sure domain is large enough
- Make sure grid resolution captures smallest features
- Remember periodic state space dimensions (angles)

Target set
- Built-in functions available to create simple shapes
- Arbitrary functions can be defined using the grid
  - g.xs{i} in this context represents i\textsuperscript{th} state

Time horizon
- dt and tau determine what $t$ is stored for $V(t, x)$
- Time discretization for computation is determined automatically to ensure numerical stability

Vehicle parameters (Dubins car’s speed and max turn rate)

Reach or avoid? (min for reach, max for avoid)
ODE model of system
• Implemented as classes, found in the DynSys folder
• Implementing extra models is relatively simple

Pack parameters and solve PDE
• \textit{minWith} parameter determines whether reachable sets or tubes are computed
• Solution is stored in data, which is a \((n + 1)\)-dimensional array

```matlab
% Pack problem parameters
72  
73  % Define dynamic system
74  % obj = DubinsCar(x, wMax, speed, dMax)
75  dCar = DubinsCar([0, 0, 0], wMax, speed);  %do dStep3 here
76  
77  % Put grid and dynamic systems into schemeData
78  schemeData.grid = g;
79  schemeData.dynSys = dCar;
80  schemeData.accuracy = 'high';  %set accuracy
81  schemeData.uMode = uMode;
82  %do dStep4 here
83  
84  % additive random noise
85  %do Step8 here
86  %HJIextraArgs.addGaussianNoiseStandardDeviation = [0; 0; 0.5];
87  % Try other noise coefficients, like:
88  % [0; 0]; % Noise on x state
89  % [0.2, 0, 0; 0.2, 0; 0; 0, 0.5]; % Independent noise on all states
90  % [0.2; 0.2; 0.5]; % Coupled noise on all states
91  % (zeros(size(g.xs(1))); zeros(size(g.xs(1)))); (g.xs(1)+g.xs(2))/20; % State-dependent noise
92  
93  % If you have obstacles, compute them here
94  
95  % Compute value function
96  
97  %HJIextraArgs.visualization = true;  %show plot
98  %HJIextraArgs.figureNum = 1;  %set figure number
99  %HJIextraArgs.deleteLastPlot = true;  %delete previous plot as you update
100  
101  % [data, tau, extraOuts] = ...
102  % HJIPDE_solve(data0, tau, schemeData, minWith, extraArgs)
103  [data, tau2, ~] = ...
104  HJIPDE_solve(data0, tau, schemeData, 'minWith', HJIextraArgs);
```