Uncertainties in Planning and MDP

1. Introduction

This report covers an implementation of real-time dynamic programming to solve the Race Track problem with uncertainties. Since the heuristic evaluation function has a major influence on the performance, different heuristic evaluation functions are tested.

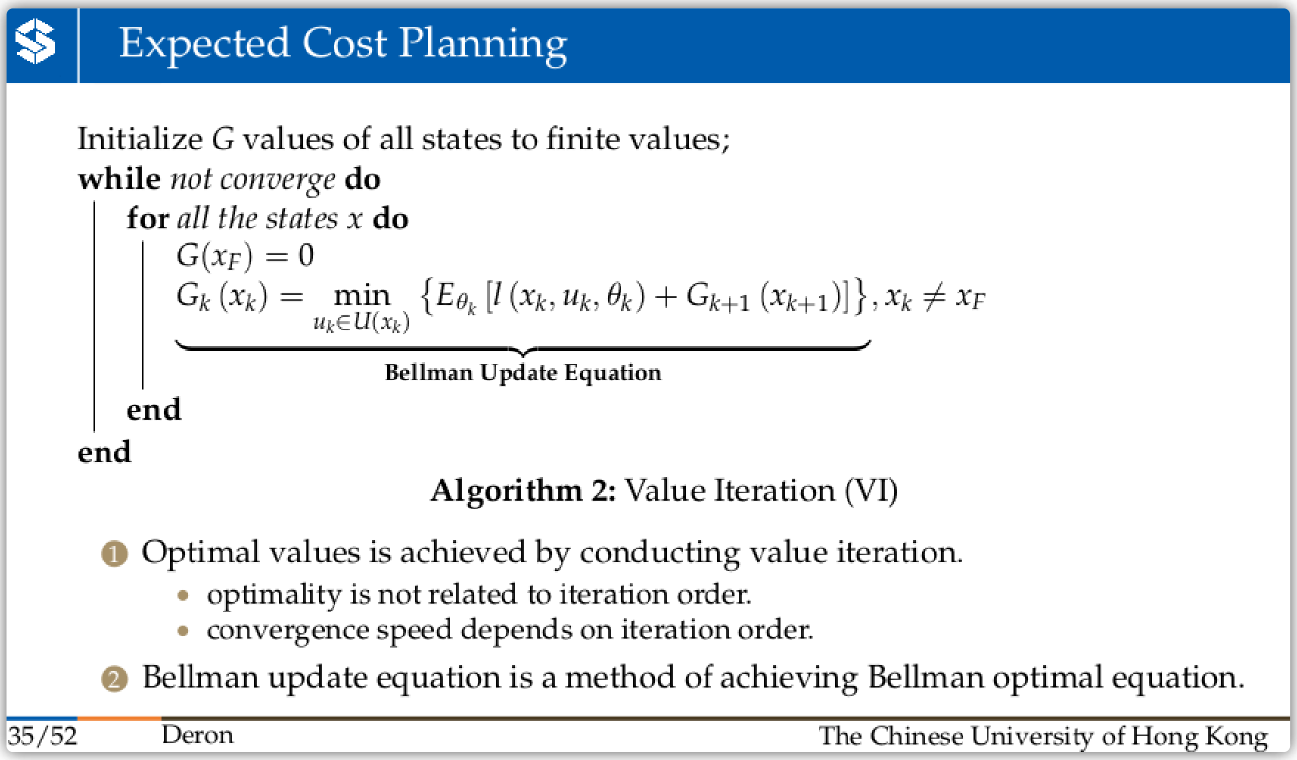
2. Fundational Concepts

2.1 Planning with Uncertainties

- Introduce two decision-makers to model the generation of uncertainties
  - Robot performans planning based on fully known states and perfect execution(model-based)
  - Nature adds uncertainties to the execution of plans made by robot, which is unpredictable to the robot.

2.2 Dynamic Programming with Uncertainies

- **Minimax Cost Planning**
- **Expected Cost Planning**
  - Probabilistic model, a specific execution maybe not optimal
  - Expected-case analysis, require distribution of uncertainties



2.3 Real-Time Dynamic Programming

The real-time dynamic programming(RTDP)[Barto et al., 1993] Algorithm is an asynchronous DP approach that updates sates encountered during heuristic-based MDP simulations. RTDP samples trajectories by greedy search and only those states will be updated.

One Key advantage of RTDP is that it may not need to explore all states and can focus on more relevant states.

Algorithm 1: RTDP

begin
  // *Initialize* $\hat{V}_h$ *with admissible value function*
  $\hat{V}_h := V_h$
  while *convergence not detected and not out of time* **do**
    $depth := 0$
    visited.CLEAR() // *Clear visited states stack*
    Draw s from $\mathcal{I}$ at random // *Pick initial state*
    while $(s \notin \mathcal{G}) \wedge (s \neq null) \wedge (depth < max\text{-}depth)$ **do**
      $depth := depth + 1$
      visited.PUSH(s)
      $\hat{V}_h(s) :=$ UPDATE($\hat{V}_h, s$) // *See (2) & (3)*
      $a :=$ GREEDYACTION($\hat{V}_h, s$) // *See (4)*
      $s :=$ CHOOSENEXTSTATE(s, a) // *See (5)*

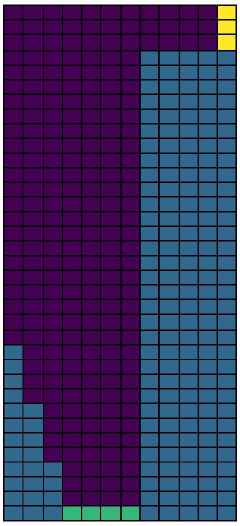
    // *The following end-of-trial update is an optimization*
    // *not appearing in the original RTDP*
    while $\neg$*visited*.EMPTY() **do**
      $s :=$ *visited*.POP()
      $\hat{V}_h(s) :=$ UPDATE($\hat{V}_h, s$)

  return $\hat{V}_h$
end

3. Experiments

3.1 Problem Statements

Real Time Dynamic Programming



Grid Map

1. $X = \{(x, y) \mid 0 \leq x \leq 11, 0 \leq y \leq 34\}$
2. $X_I = \{\text{green grids}\}$
   $X_F = \{\text{yellow grids}\}$
3. $U = \{(\ddot{x}, \ddot{y}) \mid \ddot{x} \in \{0, \pm 1\}, \ddot{y} \in \{0, \pm 1\}\}$
4. $\Theta = \{\theta_1, \theta_2\}$
   - θ_1: $f(\mathbf{x}_{k+1}, \mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k \quad p_1 = 0.1$
   - θ_2: $f(\mathbf{x}_{k+1}, \mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_{k+1} \quad p_1 = 0.9$
5. $l(\mathbf{x}_k, \mathbf{x}_k, \theta_k) = -1$
6. **Find an optimal plan from X_I to X_F**

51/52 Deron The Chinese University of Hong Kong

3.2 Implementational Details

3.2.1 Heuristic Functions

- Scaled euclidean distance

```
distance = np.linalg.norm(goal_position-current_position)
g_value = distance/dist_factor
```

- Scaled euclidean distance and velocity

```
distance = np.linalg.norm(goal_position-current_position)
velocity = np.linalg.norm(current_node.vx+current_node.vy)
value = distance/dist_factor - vel_factor*velocity
```

- Dynamic euclidean distance and velocity

```
    distance = np.linalg.norm(goal_position-current_position)
velocity = np.linalg.norm(current_node.vx+current_node.vy)
if distance >= dist_thre:
    value = distance/dist_factor - vel_factor*velocity
else:
    value = distance/dist_factor - vel_factor*velocity/10
```

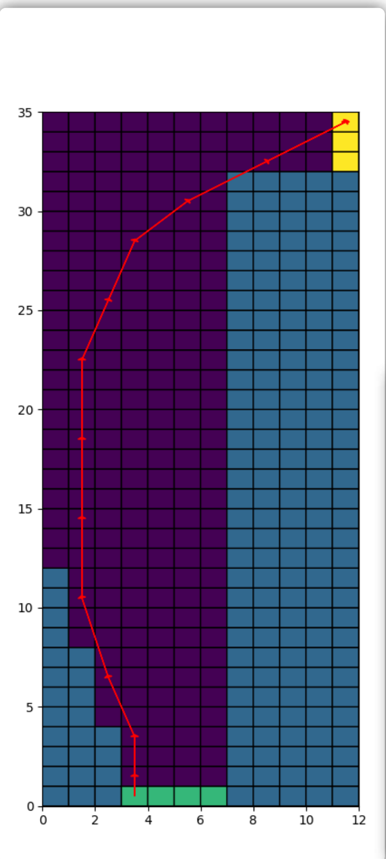
3.2.2 Exploration and Exploitation

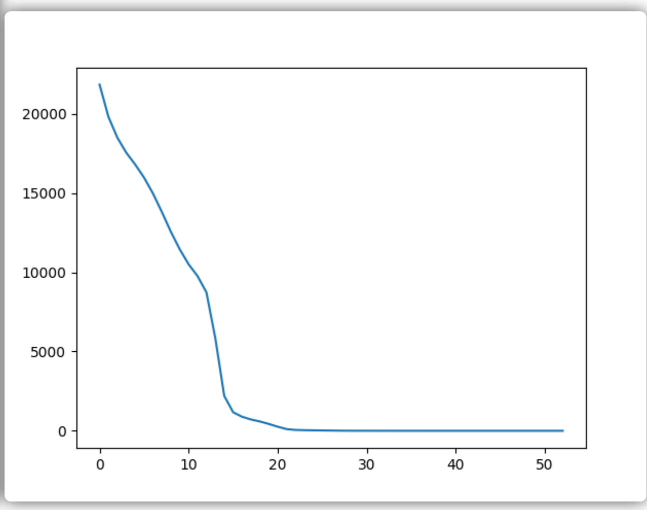
- Sample trajectories with random outcomes

```
random_prob = 0.4*np.exp(-iter_num*0.01)
if np.random.choice([0,1],p=[1-random_prob,random_prob]):
    child_key = state.next_prob_9[np.random.choice(len(value_uk))]
else:
    child_key = state.next_prob_9[np.argmin(value_uk)]
```

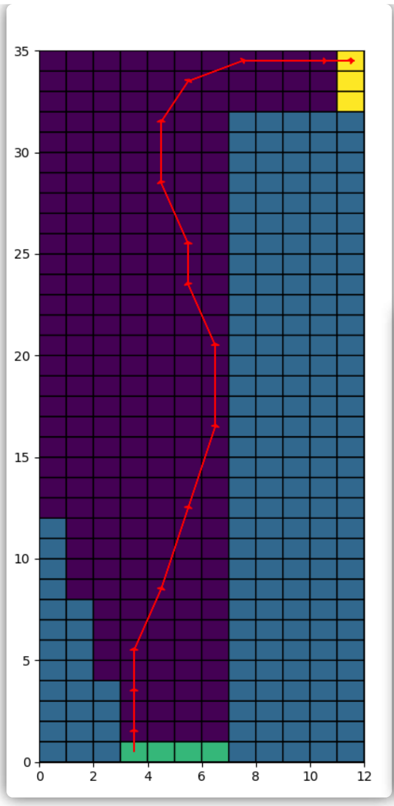
3.3 Empirical Results

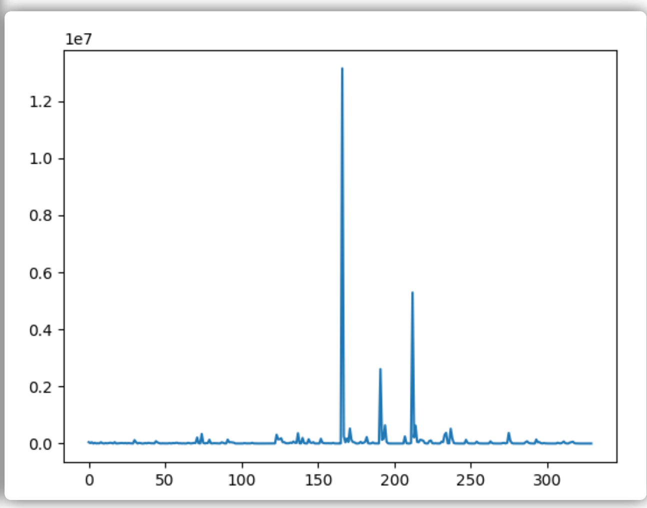
Dynamic programming without heuristic



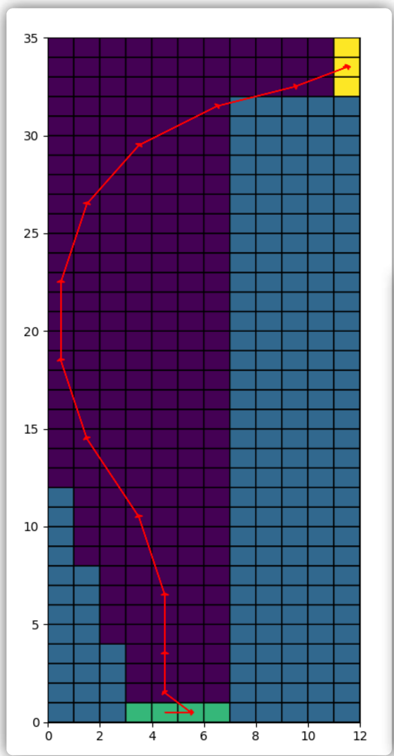


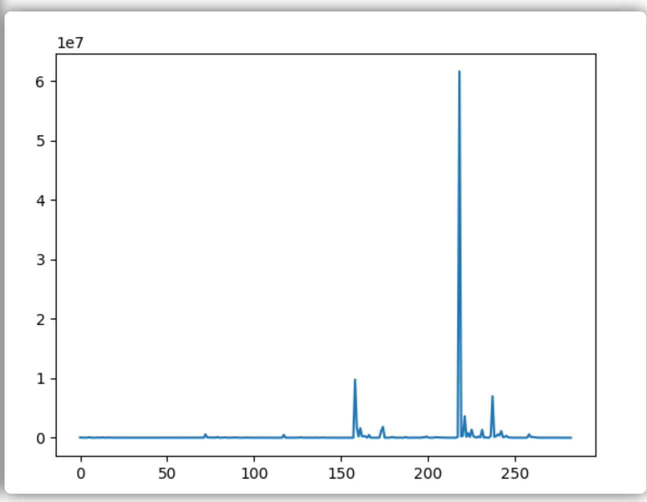
Scaled euclidean distance and velocity



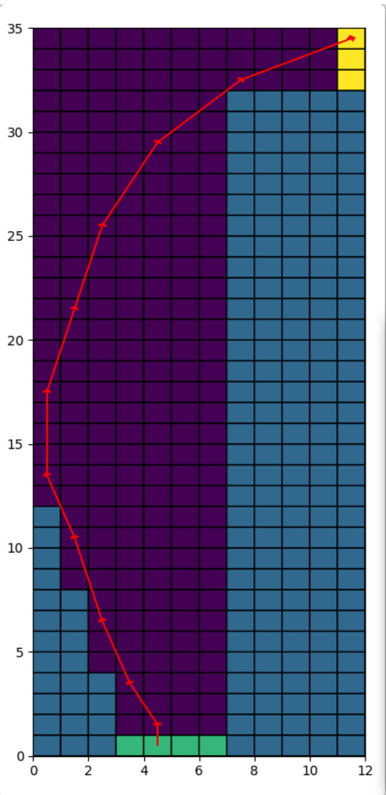


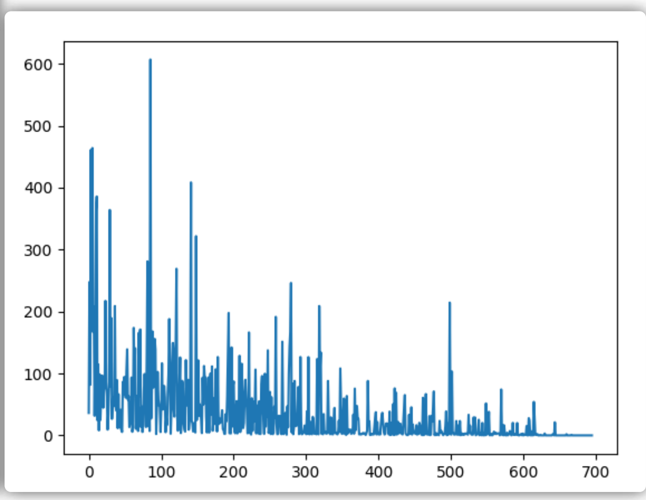
Dynamic euclidean distance and velocity





Update g_value with concern about action space





The Third heuristic has better performance

```
created a trajectory
266th iteration: 1.391774265879775e-05
running time is  48.21323323249817
0 5
1 4
3 4
6 4
10 3
14 1
18 0
22 0
26 1
29 3
31 6
32 9
33 11
```

4. Reference

Learning to Act Using Real-Time Dynamic Programmin

Course from ShenlanXueYuan