Planning Under Uncertainity

Modelling and Solving POMDP for Robot Navigation

Edwin, Niranjan, Adhithiyan

RBE — WPI

27th November 2023

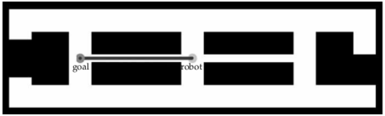
Introduction

This presentation shall cover the following

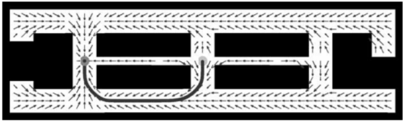
1. Background and Motivation
2. Problem Statement
3. Plan of Action
4. Tools
5. Evaluation metrics
6. Task Delegation
7. Current Environment and Proposed Framework
8. Observation and Preliminary Results
9. Future Direction

Background - Why MDPs?

- Classical path planning - an example [1]
  - Plan a sequence from start to goal - flawless execution
  - Introduce uncertainty in action - collide in narrow path

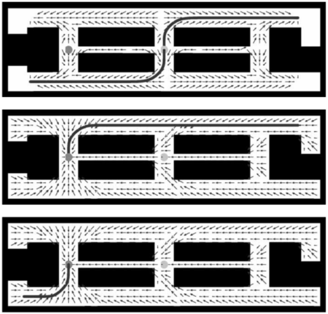


- Incorporating this uncertainty in our model - MDP
  - Markov Decision Processes - states, actions and rewards
  - Learn a strategy or policy - put agent in good value states for success
  - Example [1] - not greedy approach - risk in colliding
  - Robot rewarded for reaching goal - penalized for colliding
  - Takes a detour - more successful
  - But considers states - completely observable



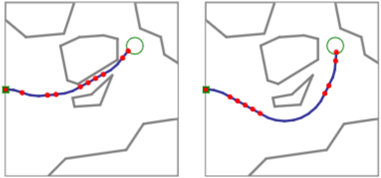
Background - Why POMDPs?

- In real world - observations of states noisy - due to imperfect sensors, etc.
- Need to incorporate uncertainty in states - POMDP
    - Partially Observable Markov Decision Processes
    - Belief states instead of states
        - Probability distribution - states robot believes to be in
    - New dimension to decision making - information gathering
    - Example [1] - robot unaware of its orientation but knows initial location
        - Good strategy - first move to a corner
        - Depending on the corner - moves accordingly



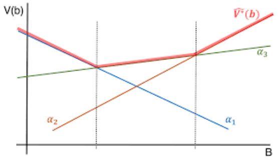
Background - Problem Statement

- Robot model - Dubin's car - to reach goal configuration in a cluttered environment
- An application - steering medical needle to required tissue [2]
  - Modeled as an MDP
  - More success by avoiding a narrow, riskier path



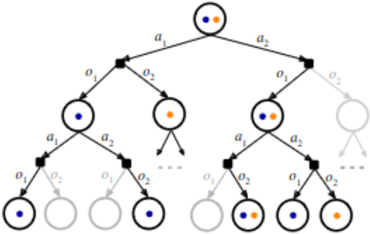
- Objective - capture the uncertainty in states also - POMDP
- Uncertainty - robot location, environment map [3]
- Solve the POMDP - a good enough policy to reach desired configurations with more success

Background - POMDP Solvers



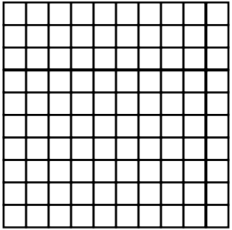
- Optimal policy - outputs best action to take for each possible belief state
  - After computing for specific horizon - piecewise continuous value function
  - Each region of belief space - specific action optimal
- POMDP as a problem is exponential in nature, deeper you go.
  - Curse of History
  - Curse of Dimensionality
- We will be evaluating some tree-based POMDP solvers to find good policy for the problem at hand

Plan of Action - DESPOT



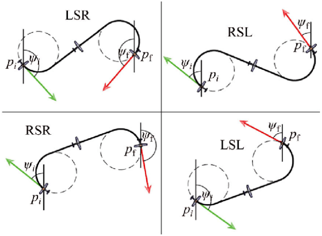
- Some solvers considered for our work
  - DESPOT [4]
  - POMCP
- DESPOT works by randomly initializing a set of scenarios and then only exploring the belief space that consists of those scenarios.
- This reduces the complexity from $O(|A|_D|Z|_D)$ to $O(|A|_D K)$ nodes, where $|A|$ and $|Z|$ are the sizes of the action set and the observation set

Current Environment



- The current environment - 10×10 grid world
- Includes various scenarios of start and stop positions
  - Each state encoded as SSOO
  - SS $\rightarrow$ Coordinate ID (79 $\rightarrow$ 8th row 10th column)
  - OO $\rightarrow$ Orientation ID (02 $\rightarrow$ 90 degrees)

Dubin's Car in Grid World



- Reaching goal configuration from start using only few turns of constant curvature (LSL, RSR, RSL, etc.) under certain conditions
- Cannot travel in reverse direction
- In the grid world
  - Turns of only $\frac{\pi}{4}$ at a time allowed
  - Steps of only one grid jump allowed
  - Cannot turn without moving forward
  - Example car facing north
  - Can only go north, northwest, or northeast by one step

Proposed Framework

- **States**
  - Each cell in the 10×10 grid - valid states
- **Actions**
  - Normal grid traversal - 5 actions: up, down, left, right, halt
  - Dubin's car - 4 actions: go straight, steer left, steer right, halt
- **Observations**
  - Agent-Environment state as perceived by the agent - noisy
- **Rewards**
  - $+40$ to state-action pairs that result in goal state
  - -1 for every other valid state-action
  - -10 if agent tried to leave grid

Tools

Tools and repositories used for this project are listed below:

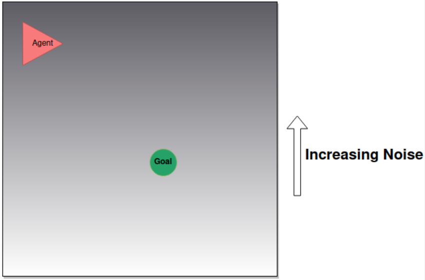
- Python_Robotics
  - Used for visualizing and planning for Dubins Car
- Pypomdp
  - Used for solving POMDP using POMCP
  - Code for visualizing and plotting the belief state
- Despot API
  - Used to solve POMDP in pomdpx format
- APPL toolkit(SARSOP)
  - Used pomdp_convert function to convert /.POMDP to /.pomdpx format

Evaluation Metrics

- Although POMDP solvers powerful for planning
  - Notorious for high computational complexity
- Plan
  - Use tree-based solvers to get approximate optimal policies
  - Comparison between POMCP, DESPOT solvers
- Parameters
  - Online Time
    - Time taken to reach the goal and end runtime
    - Necessary to make sure the solution is not intractable.
  - Task Completion
    - Consider the probability of the robot reaching the goal
    - Evaluate in multiple scenarios

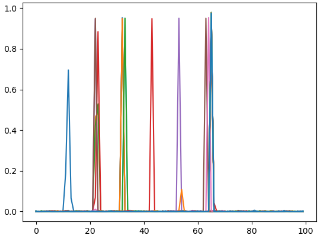
Methodology

- Problem Formulation
  - Problem was written in the /.POMDP format
  - State Transition probabilities, Observation Probabilities were defined
  - File was converted to /.POMDPX format to use different solvers
- Defining Observation Probabilities
  - Added noise distribution around neighbouring cells
  - State uncertainty decreases as we go down
- Strategy
  - Derived idea from Light-Dark method
  - Expectation - Agent moves to state with less uncertainty and gathers information

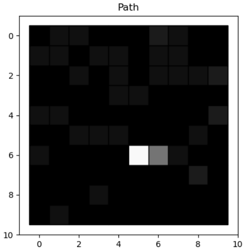


Observation - POMCP

- Agent starts at state 2 and aims to reach state 65.



((a)) Belief States

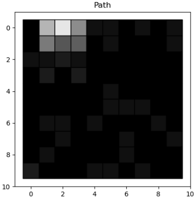


((b)) Visualization

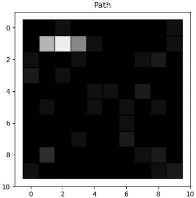
Figure: Results from POMCP

- **Trajectory**
  - '3', '13', '23', '23', '23', '22', '22', '32', '32', '32', '32', '33', '32', '32', '33', '43', '53', '63', '64', '65', '65', '65', '65', '65', '65', '65'

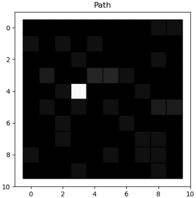
Trajectory Visualization - POMCP (sort of)



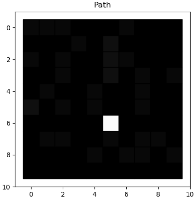
((a)) Initial Belief State



((b)) After a while



((c)) After a long while

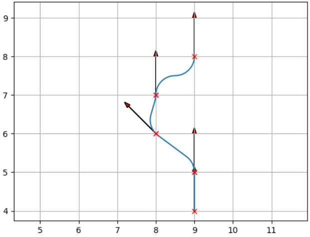


((d)) Finally...

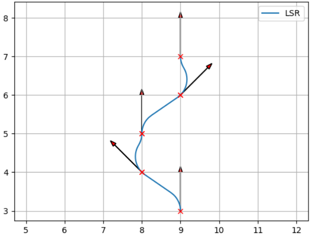
Implementation

- Problem Definition
  - Program developed to generate POMDP file
  - Both Action Uncertainity and Observation Uncertainity are encoded
- Conversion
  - The POMDP file was converted to pomdpx utilising the SARSOP pomdpx_convert file.
- Visualization
  - Extracted States and Belief from DESPOT solver

Path in Dubins



((e)) DESPOT Solver

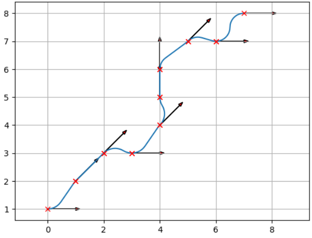


((f)) POMCP Solver

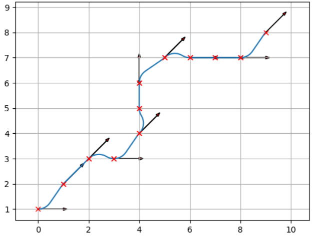
Figure: Results of start point [6,9]

- DESPOT Trajectory: s6902, s5902, s4803, s3802, s2902
- POMCP Trajectory: '6902', '5803', '4802', '4802', '3901', '2902'

Path in Dubins



((a)) DESPOT Solver



((b)) POMCP Solver

Figure: Results of start point [9,0]

- DESPOT Trajectory: s9000, s8101, s7201, s7300, s6401, s5402, s4402, s3501, s3600, s3700, s3800, s2901
- POMCP Trajectory: '9000', '8101', '8101', '8101', '7201', '6202', '6202', '5301', '5400', '5500', '5600', '4701', '3801', '2901'

Results - Conclusion

- 1st Instance - Start(6,9)
  - DESPOT - 5 steps
  - POMCP - 6 Steps
- 2nd Instance - Start(9,0)
  - DESPOT - 12 steps
  - POMCP - 14 Steps
- POMCP tends to halt at various positions - increased number of steps and computation time.
- Time taken by POMCP(mins) to complete the task was exceedingly large than DESPOT(secs) Despot takes $\leq$ 1 min whereas POMCP takes $\geq$ 15 mins

Task Completion - Evaluation

- Translate from grid traversal to Dubins Car (✓)
- Visualize implemented model in DESPOT Solver instead of POMCP - trajectories (✓)
- Define Gaussian Noise distribution for more exploration (✓)
- Take Action uncertainty into account (✓)
- Extract Belief states from DESPOT after each step (✓)
- Figure out why Belief State is not making sense.

References

S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

R. Alterovitz, T. Siméon, and K. Goldberg, "The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty," in *Robotics: Science and Systems III*. The MIT Press, 01 2008. [Online]. Available: https://doi.org/10.7551/mitpress/7830.003.0031

M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable markov decision processes in robotics: A survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, 2023.

N. Ye, A. Somani, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *The Journal of artificial intelligence research*, vol. 58, pp. 231–266, 2017.