Review MDP, RL and UAV system modeling under uncertainty

Minjun Kim-xS3d master candidate

Friday September 9th, 2016

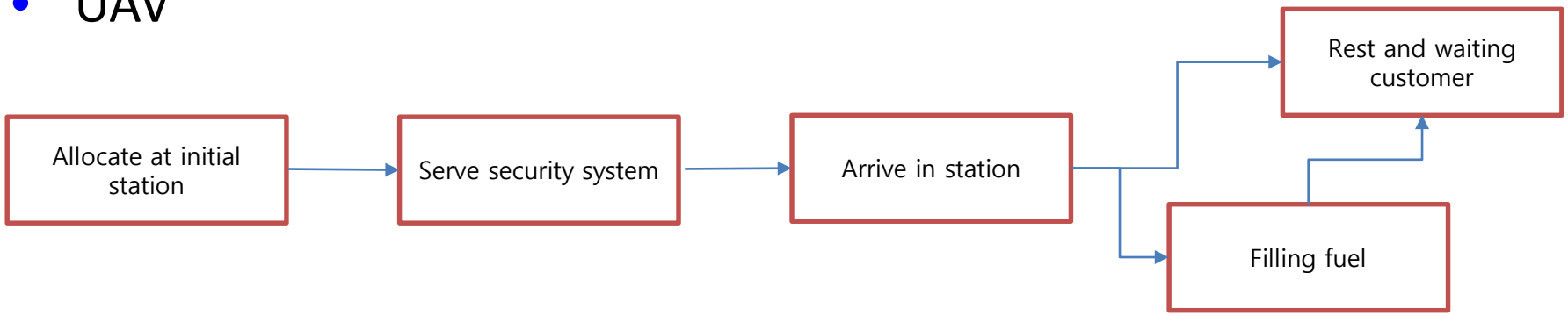
Tables of contents

Introduction (UAV)

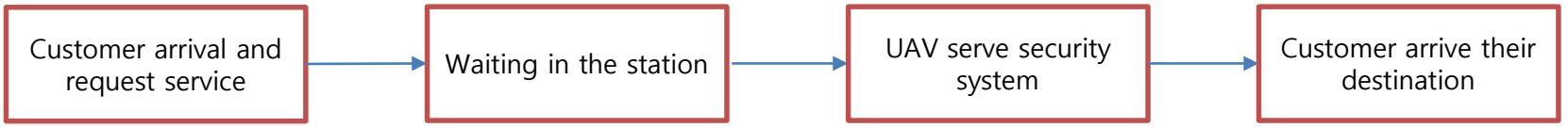
- These days, Interest for UAV technologies have been growing
- Applications of UAV
  - Package delivery, Fire monitoring, Target tracking, Surveillance, Tourism, Electronic message relaying
- Among various UAV application, we use UAV serve security service for customer
- A key question for such a large scale system : what resources should be used at what time?

Introduction (System architecture)

- UAV

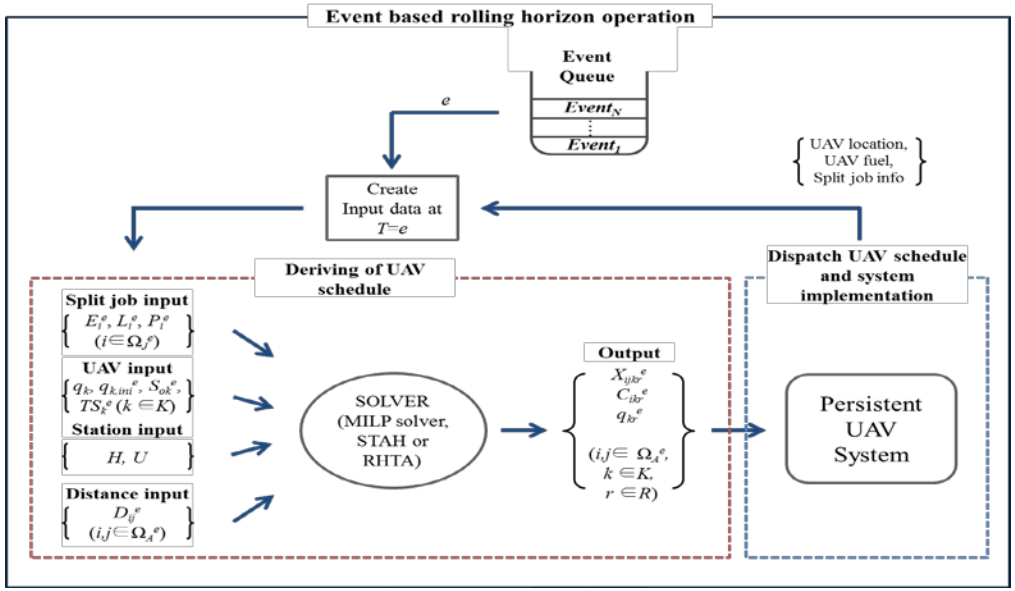


- Customer



Introduction

- Previous solution approach
  - Deterministic with rolling horizon approach

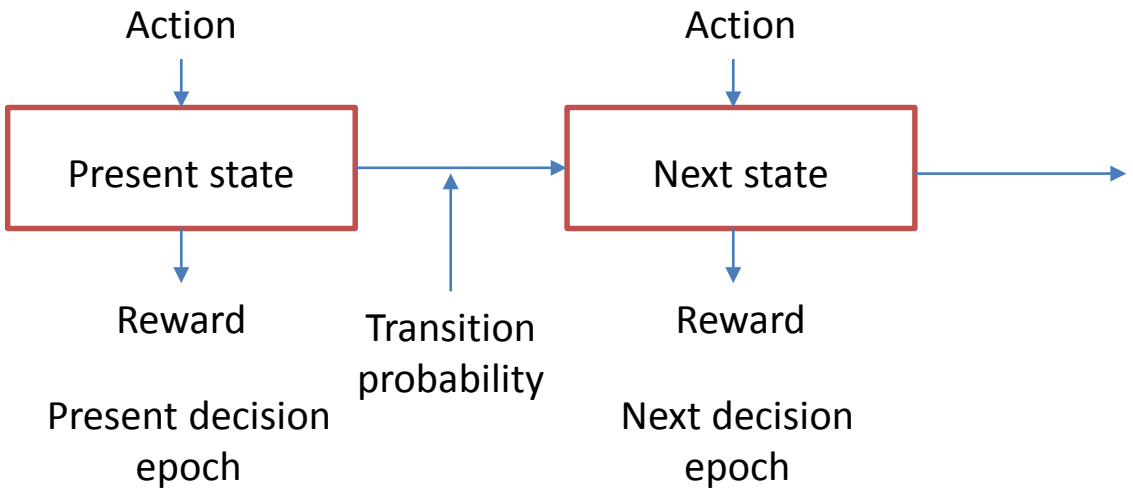


- Suggest solution approach
  - MDP & RL for stochastic environment

Background(MDP)

- Mathematical framework for posing sequential decision problems, including multi-agent problems

- Key ingredients in Markov decision process
  - A set of system states
  - Available actions
  - Decision epochs
  - Transition probability
  - Reward



Background(MDP quantities)

- MDP`s goal is find optimal policy which maximize expected reward
  - P: transition function, R:reward function, V: value function

$$\pi(s) := \arg\max_a \left\{ \sum_{s'} P_a(s,s') \left(R_a(s,s') + \gamma V(s') \right) \right\}$$

- Bellman equation
  - Value function can be a expressed as a Bellman equation

$$V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$$

  - Bellman equation is solved by value iteration and policy iteration

Background(MDP-example)

- Grid world is the most famous example in MDP

			+1
			-1
Start			

Possible action:

Up, Down, Left, Right

Goal: maximize reward

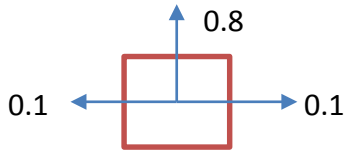
Deterministic Solution

Up, Up, Right, Right, Right
Probability:1

If actions are unreliable

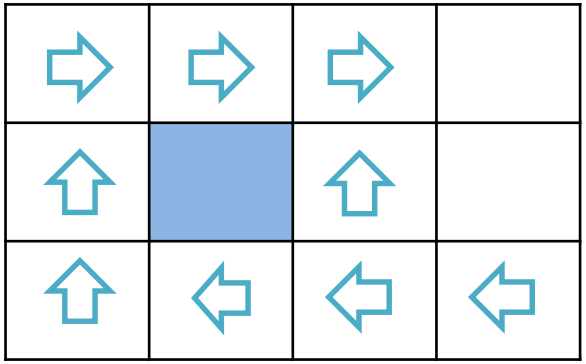
Probability 0.8 to go to intended cell

Probability of reaching the goal state by executing Up, Up, Right, Right, Right: 0.32776



Background(MDP-example)

- Solution for stochastic grid world
  - Input: R(s)=-0.04, no discount factor



Policy

0.812	0.868	0.918	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

Value

Value iteration

- Value iteration is an algorithm for computing an optimal policy
- Idea of VI: Calculate the utility of each state and then use the state utility to select an optimal action in each state

Initialize V arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Pseudocode for VI

Reinforcement learning: An introduction
Richard S. Sutton and Andrew G. Barto

Policy iteration

- Idea of PI:
  - Policy evaluation: start with a random policy and then calculate utilities based on if that policy were executed
  - Policy improvement: calculate a new policy based on computed utilities
  - Iterate until the policy does not change

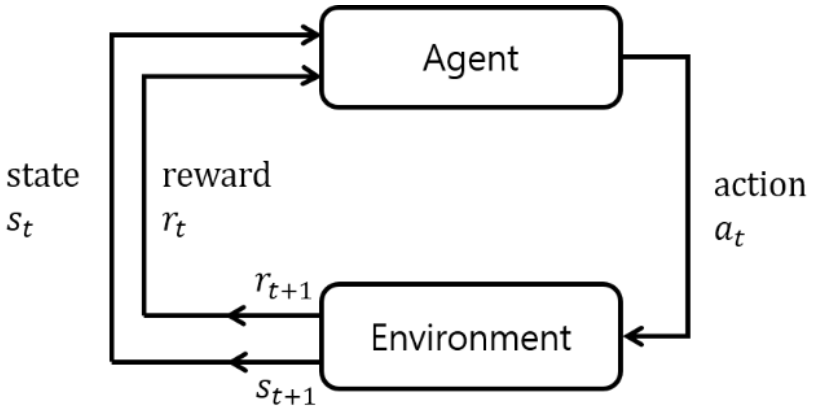
1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $b \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$
   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop; else go to 2

Pseudocode for PI

Background RL



- Basic idea of RL
  - Receive feedback from the rewards
  - Agent`s utility is defined by the reward function
  - Learning`s goal is maximize expected rewards
  - All learning is based on observed samples of outcomes

Example of RL

- Robot Learns to Flip Pancakes

  - https://www.youtube.com/watch?v=SH3bADiB7uQ&index=2&list=PL5nBAYUyJTrM48dViibyi68urttMlUv7e
- Stanford Autonomous Helicopter

  - https://www.youtube.com/watch?v=VCdxqn0fcnE&list=PL5nBAYUyJTrM48dViibyi68urttMlUv7e&index=3

Q-learning

- Model-free RL: use the experience to directly learn a value function

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

(Learning rate) (New sample estimate)

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$
    $s \leftarrow s'$;
  until s is terminal

Pseudocode for Q-learning

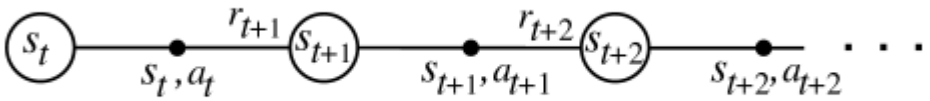
Q-learning

- Q-learning convergence
  - $\lim_{t\to\infty} Q(s,a) = Q^*(s,a)$
  - All states are infinitely visited and each action is executed an infinite number of times
  - $\alpha_s = \frac{1}{number\ of\ visits\ to\ state\ s}$
  - $\sum_{i=0}^{\infty} \alpha_s = \infty$ but $\sum_{i=0}^{\infty} \alpha_s^2 = \infty$

- Exploration/ Exploitation problem
  - Needs to search all state and find the optimal policy for current environment
  - ε-greedy action selection
    - Choose a greedy action with (1- ε) and a random action with probability ε

SARSA

- SARSA (State - Action - Reward – State - Action)



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big]$$

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r + \gamma Q(s', a') - Q(s, a)\big]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

Pseudocode for SARSA

Other RL techniques

- POMDP (partially observable MDP)
  - Agent indirectly observes environment
  - Set of observations and conditional observation probabilities are added to MDP

- Functional approximation
  - Approximate Q-value
  - $Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$

- State reduction
  - Reduce the state space
  - Aggregate the states which has similar property

Compare with MDP toolbox

- Compare my code with Matlab MDPtoolbox
  - Toolbox is made by the decision team of the Applied Mathematics and Computer Science Unit of INRA Toulouse (France)
  - I made MDP, RL code for (3X4) grid world

3	5	8	11
2		7	10
1	4	6	9

- Value iteration, Initial value: 0 for all state

	1	2	3	4	5	6	7	8	9	10	11
1	0.7053	0.7616	0.8116	0.6553	0.8678	0.6114	0.6603	0.9178	0.3879	-1	1
2	1	1	3	4	3	4	1	3	4	0	0

- Policy iteration, Initial policy: right for all state, set initial value as 0

	1	2	3	4	5	6	7	8	9	10	11
1	0.7053	0.7616	0.8116	0.6553	0.8678	0.6114	0.6603	0.9178	0.3879	0	0
2	1	1	3	4	3	4	1	3	4	0	0

Compare with MDP toolbox

- Result of Q-learning
  - R(s)=-0.04, Discount factor=0.9 Iterations: 3000000
  - Learning rate: (1/(n(s,k)^(5/9)))
    (n(s,k): number of visits for specific state and action)

0.570051	0.699882	0.852487	0.604996	1.043642	0.741768	0.937864	1.243464	0.564302	-0.48766	1.512423

  - Learning rate: (1/(n(s,k)^(2/3)))

0.568047	0.698522	0.850977	0.600597	1.044054	0.740068	0.942078	1.243836	0.541587	-0.48853	1.511193

  - Result of MDP toolbox

0.551609	0.686304	0.839492	0.547349	1.030339	0.676864	0.880546	1.230746	0.444032	-0.5002	1.497893

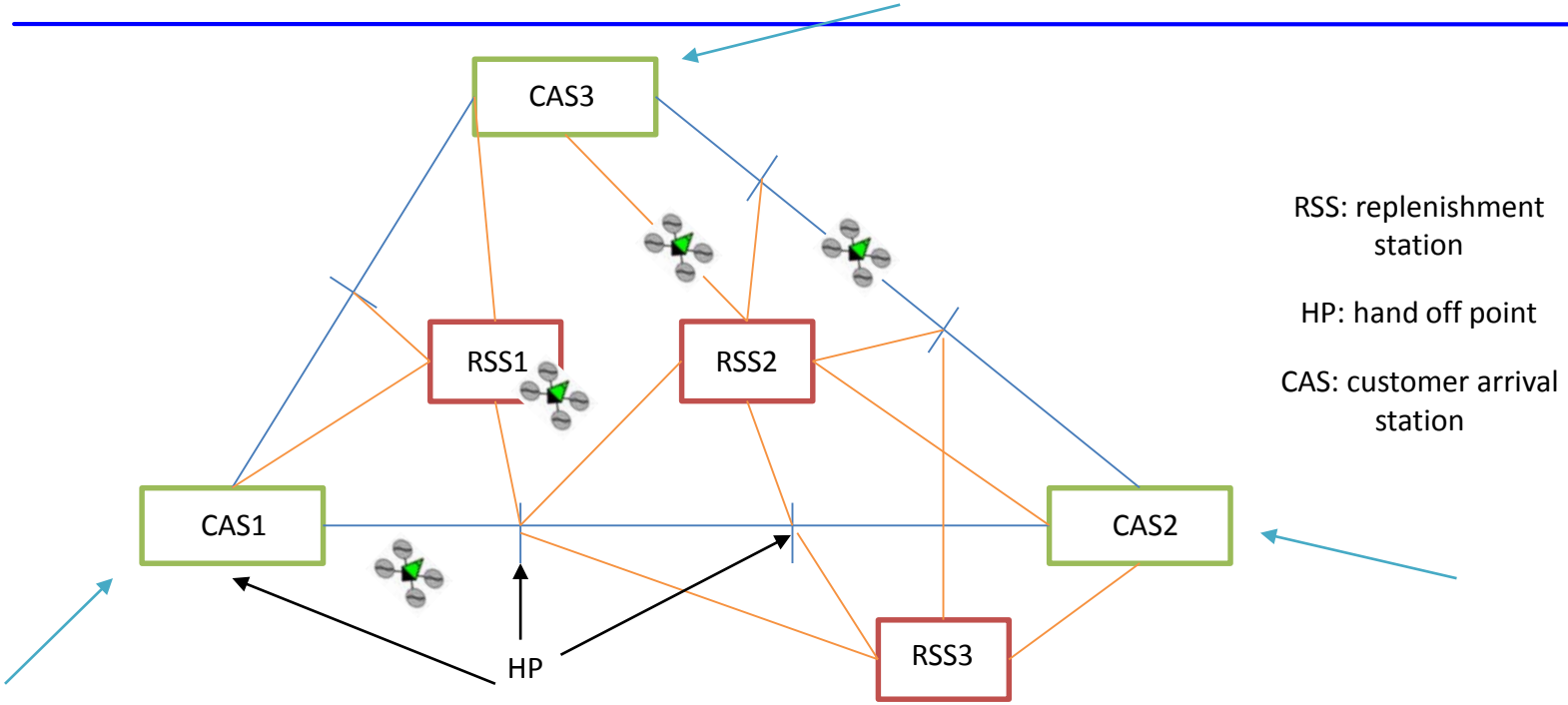
  - Result of SARSA

0.450477	0.581973	0.732382	0.422009	0.926176	0.556177	0.772051	1.134099	0.326315	-0.59598	1.404643

Compare with MDP toolbox

- In case of value iteration and policy iteration, result is same as MDP toolbox also it can rapidly find a optimal policy
- In case of RL, it spend time for converge optimal value
- As number of visit to state and action pair is increased, converge fast
- Result is influenced by learning rate
  - As learning rate is closed to (1/(n(s,k)^(1/2))) it converge fast

UAV system description



- Customer arrived to CAS
- Each line section has two jobs (both directions)
- UAV rest and wait customer in HP
- Split job is divided into P-job (RSS↔HP, orange) and C-job (security service, blue)
  - In this example 28 P-jobs and 16 E-jobs, 3 RSS, 3 CAS, 8 HP

Problem description of UAV system

- Objective
  - Maximize customer`s split job (E-job) serve rate
- Constraints
  - UAV`s fuel limitation
  - Total number of customer in the system
- Assumption
  - Fixed design and fixed resource in the system
  - Customer trajectory is fixed, based on their request
  - If UAV is not prepared customer is waited until UAV is prepared
  - UAV`s failure is not considered
  - Each UAV serve 1 customer at a time
  - There are sufficient charging machine in each RSS

Generic State of UAV system

- Index for UAV: i , Number of UAV: n
- Index for C- job: j, Number of C- job: m
- Index for jobs and RSS and HP: k, Number of jobs and RSS and HP: p
- Location of UAV_i : $l_i \in \{split\ job\ or\ RSS\ or\ HP\ k\}$
- Fuel level of UAV_i: $f_i \in \{\Delta f, 2\Delta f, 3\Delta f, \dots, F_{max} - \Delta f, F_{max}\}$
- Number of waiting customer for each E-split job:

$$w_j \in \{0,1,..,c\}, \quad \sum_{j=1}^{m} w_j \leq h$$

- Representation of general state

S=$\{(l_1, f_1), (l_2, f_2),\dots, (l_n, f_n), w_1, w_2,\dots, w_m\}$

UAV system modeling

- Event: Customer arrival, Finish split job, Finish fuel replenishment
  - If event is occurred, transition is happened

- Action: UAV *i* move to direction *k* ,UAV *i* filling fuel, UAV *i* stop, Increase or decrease number of waiting customers in E-job

- If E-split job is served, give a reward

- Reward function can be represented as
  - $R(s,a) = R_j * n_E$ ($n_E \in \{0,1\}$ indicates if agent is served E-job)

    R_j: Associated reward for E-job j

Future research plan

- Formulate generic MDP model
- Solve generic MDP model using value iteration or policy iteration
- Suggest reduction method to reduced the complexity of the problem
- Compare optimal solution with deterministic rolling horizon approach`s optimal solution