

Markov Decision Process for Train Conflict Resolution


Presented by SungYub Kim

Management Science/Optimization Lab
Department of Industrial Engineering
Seoul National University

April 7, 2017

Contents and References

- Train Conflict Resolution Problem
 1. MIP modeling for Centralized Train Conflict Resolution Problem¹
 2. Dynamic and Autonomous train driving
 3. Markov Decision Process(MDP)
- Algorithms for MDP
 1. MDP Solution Concept
 2. Policy Iteration Algorithm
 3. Value Function Approximation
- Appendix

¹Y.H. Min and M.J. Park and S.P. Hong and S.H. Hong, "An appraisal of a column-generation-based algorithm for centralized train-conflict resolution on a metropolitan railway network." *Transportation Research Part B*, 2011. 

Train-Conflict Resolution Problem

- What is train-conflict resolution problem?

1. A *train timetable* is the set of *station*, *arrival time*, *departure time*, *track assignment* for all trains and the stations which they visit over a predetermined time horizon.

2. A *train-conflict* is a situation in which two or more trains claim resources in an infeasible manner, namely, in the way that violates a safety regulation.(especially safeheadway)

3. A *train-conflict resolution problem* has following inputs and outputs.

Input : A railway network, a timetable, and a train-conflict

Output : A conflict-free timetable with the minimum total weighted deviation with respect to the original timetable which satisfies the constraints on the earliest departure times, the headway times, the transit times, and the dwelling times.

Constraints for conflict-free timetable

- 1 Track Assignment
- 2 Dwelling Pattern selection
- 3 Upper and Lower bound of transit time
- 4 Dwelling time requirements
- 5 Train-dispatch sequencing
- 6 Earliest departure times
- 7 Intra-track headway safety
- 8 Inter-track headway safety
- 9 Deviations from the original timetable

현기술 상태의 취약점과 대안

- 시스템 운영의 보수성

1. 기존의 지상 중심 제어시스템은 정보를 교환하는 과정에서 열차 속도, 위치 등의 정보를 얻는데 있어서 지연 및 오차가 발생함.
2. 따라서, 열차 간 안전시격을 보수적으로 확보하며, 이는 효율의 감소를 발생시킴.

- 시스템 확장의 한계점

1. 기존의 중앙집중식 운영 방식의 경우, 노선 확장이나 차량의 증편 등의 시스템 변경 사항이 발생했을 때, 이를 대처하는 능력이 전체 시스템 규모에 영향을 받음
2. 또한, 다수의 운영 주체와 교통 수단을 포함하는 대규모 네트워크를 운영하는데 한계가 있음.

- 대안

1. 중앙집중식 운영 방식에 비해 대규모 네트워크 운영에 유리한 다중 에이전트 모델링 고려.
2. 변화하는 상황을 반영하기 위해 동적 모델링 고려.

Markov Decision Process(MDP)

Markov Decision Process는 동적인 환경 속에서 장기적인 보상을 고려하는 에이전트가 의사 결정을 내리는 상황을 모델링합니다. 이 때, 에이전트는 매 시점마다 하나의 상태에 놓여있고, 이 때마다 하나의 행동을 선택할 수 있으며, 선택할 때마다 보상을 받고 다음 상태로 넘어갑니다.

Definition 3.1

Markov Decision Process

- 1 \mathcal{S} is a finite set of *states*,
- 2 \mathcal{A} is a finite set of *actions*,
- 3 $T_{s,s'}^a = P[S_{t+1} = s' | S_t = s, A_t = a]$ is the probability that action a in state s at time t will lead to state s' at time $t+1$,
- 4 $\mathcal{R}_a^s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$ is the expectation *reward* received when state s agent chooses action a ,
- 5 $\gamma \in [0, 1]$ is the *discount factor*, which represents the difference in importance between future rewards and present rewards.

MDP Example-Frozen Lake

MDP로 모델링이 가능한 상황 중 단순한 상황인 Frozen Lake는 다음과 같습니다. 먼저 에이전트는 4×4 의 지도 중 Start에서 시작합니다(State). 에이전트는 상하좌우 중에서 하나를 선택할 수 있으며(Action), 선택한 방향의 반대 방향을 제외한 나머지 방향 중에서 각각 $1/3$ 의 확률로 이동하게 됩니다.(State Transition Probability). 에이전트는 Goal에 도착했을 때, 보상 1을 받고 종료하며, Hole에 도착했을 때는 보상 0을 받고 종료합니다.(Reward).

S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G



S: Start
F: Frozen Surface
H: Hole
G: Goal

MDP Solution Concept

MDP의 최적해를 구하는 알고리즘을 찾기 위해서는 먼저 MDP의 해의 개념과 가치를 정의할 필요가 있습니다.

Definition 4.1

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s] \quad (1)$$

이를 통해 policy가 고정되었을 때, 각 상태와 행동의 가치를 다음과 같이 나타낼 수 있습니다.

Definition 4.2

A value of state s given policy π , $v_\pi(s)$, is the expected return starting from state s , and then following policy π

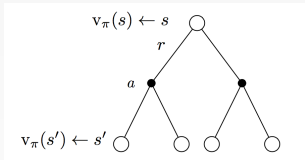
$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \quad (2)$$

Also, a value of action a given state s , $q_\pi(s, a)$, is the expected return starting from state s , taking action a , and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a] \quad (3)$$

MDP Solution Concept-cont.

앞에서 정의한 $v_\pi(s)$ 와 $q_\pi(s, a)$ 는 다음과 같은 관계를 갖습니다.



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a), \quad q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} T_{s,s'}^a v_\pi(s') \quad (4)$$

그리고 이를 정리하면 다음과 같습니다.

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} T_{s,s'}^a v_\pi(s') \right) \quad (5)$$

이 때, 각 k 번째 알고리즘 단계마다 k 번째 $v_\pi(s')$ 를 기반으로 $k+1$ 번째 $v_\pi(s)$ 를 갱신하면서 주어진 policy하에서 각 상태의 가치를 구하는 알고리즘을 **iterative policy evaluation** 알고리즘이라고 합니다. 또한, 이 반복 알고리즘은 **contraction mapping theorem**에 의해 실제 v_π 로 수렴합니다.(Appendix)

Policy Iteration Algorithm

MDP의 최적해를 구하기 위한 Policy Iteration 알고리즘은 두 단계를 반복하면서 MDP의 최적 policy를 찾습니다.

첫 번째는 초기 policy의 policy evaluation을 하는 것입니다. 앞에서 구한 알고리즘을 이용해서 $v_\pi(s)$ 를 추정하면 4 를 통해 $q_\pi(s, a)$ 또한 구할 수 있습니다. 이를 기반으로 각 상태에서 가장 좋은 행동을 취하여 policy를 개선할 수 있습니다.(policy improvement) 이렇게 개선된 policy는 모든 상태에서 이전의 state value $v_{\pi'}(s)$ 보다 더 좋은 state value $v_{\pi'}(s)$ 를 가집니다.

$$\begin{aligned}
 v_\pi(s) &\leq q_\pi(s, \pi'(s)) = E_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
 &\leq E_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
 &\leq E_{\pi'}[R_{t+1} + \gamma R_{t+1} + \gamma^2 q_\pi(S_{t+2}, \pi'(S_{t+2})) | S_t = s] \\
 &\leq E_{\pi'}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s] = v_{\pi'}(s)
 \end{aligned} \tag{6}$$

따라서 이러한 policy evaluation-policy improvement를 반복하던 중에 더 이상 개선할 수 없게 되면 이는 현재 policy가 최적임을 의미합니다.

Policy Iteration Algorithm for Frozen Lake

이제 policy iteration 알고리즘을 앞에서 모델링한 frozen lake 상황에 적용하여 최적 policy를 계산하겠습니다. Policy iteration 알고리즘 수행 동안 5번의 policy evaluation-improvement 반복을 겪었으며 아래 그림은 각각의 반복 단계에서의 $v_\pi(s)$ 와 greedy policy를 나타낸 것입니다.

0	0	0.02	0.01	0	0.02	0.06	0.05	0.02	0.03	0.07	0.05	0.07	0.06	0.07	0.06	0.07	0.06	0.07	0.06
0	0	0.05	0	0	0	0.09	0	0.05	0	0.11	0	0.09	0	0.11	0	0.09	0	0.11	0
0	0	0.16	0	0	0.15	0.25	0	0.10	0.22	0.29	0	0.14	0.25	0.29	0	0.14	0.25	0.30	0
0	0	0.48	0	0	0.25	0.58	0	0	0.36	0.63	0	0	0.37	0.64	0	0	0.38	0.64	0
L	D	R	U	U	R	R	U	L	U	R	U	L	U	L	U	L	U	L	U
L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
L	D	L	L	D	D	L	L	U	D	L	L	U	D	L	L	U	D	L	L
L	D	R	L	L	R	D	L	L	R	D	L	L	R	D	L	L	R	D	L

한편 policy iteration 알고리즘의 복잡도는 m 개의 action과 n 개의 state가 있을때, $O(mn^2)$ 입니다.(action로 자신을 제외한 나머지 state에 대해서 계산) 따라서, 이러한 알고리즘은 상태 수가 많거나 상태의 항목 중 하나가 실수 범위를 가지게 되면 적용할 수 없는 알고리즘이 됩니다.

Value Function Approximation

따라서 이러한 policy iteration 알고리즘의 한계를 극복하기 위해 policy iteration의 policy evaluation 단계를 근사적으로 만드는 것을 고려해볼 수 있습니다. 이러한 발상에서 출발한 기법들을 Value Function Approximation이라고 합니다. 또한, 이러한 알고리즘들은 에이전트가 MDP의 모든 정보를 알지 못하더라도 자신의 policy에 대한 Oracle-call을 할 수 있다면 이를 기반으로 최적 policy를 찾을 수 있습니다. 대표적인 알고리즘으로는 $q_\pi(s, a)$ 를 w 로 parameterize된 $Q(s, a; w)$ 로 근사하는 Q-learning 기법이 있습니다. Q-learning 알고리즘은 다음과 같습니다.

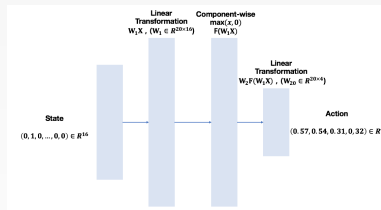
Algorithm 6.1

Q-learning

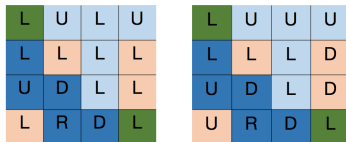
- Initialize approximate value function $Q(s, a; w)$ for all s, a
- For episode $i = 1, \dots, M$ do
 - Initialize state s_1
 - For time $t = 1, \dots, T_i$ do
 - With probability ϵ select a random action a_t
 - Otherwise select $a_t = \max_a Q(s_t, a; w)$
 - According to selection oracle gives reward r_t and new state s_{t+1}
 - Set $y_{s_t} = \begin{cases} r_t & \text{for terminal } s_{t+1} \\ r_t + \gamma \max_{a'} Q(s_{t+1}, a'; w) & \text{for non-terminal } s_{t+1} \end{cases}$
 - Perform a gradient descent step on $(y_{s_t} - Q(s_t, a_t; w))^2$
- Return $Q(s, a; w)$

Q-learning Algorithm for Frozen Lake

앞의 실험과 마찬가지로 Q-learning 알고리즘을 Frozen Lake 상황에 적용볼 수 있습니다. 여기서는 approximation 함수를 다음과 같이 정의했습니다.



이렇게 구성한 네트워크를 기준으로 2000번의 에피소드를 실행한 결과 아래의 우측과 같은 policy를 얻었습니다. 좌측은 앞선 policy iteration을 통해 얻은 optimal policy 입니다. 진하게 색칠된 경로는 에이전트가 각각의 policy를 따를때 겪게 되는 state들입니다.



Convergence of policy evaluation

Appendix에서는 앞에서 언급한 Policy evaluation 알고리즘이 실제 v_π 에 수렴함을 Contraction Mapping Theorem을 통해 보이겠습니다. 다음은 γ -contraction mapping의 정의와 Contraction Mapping Theorem입니다.

Definition 7.1

An operator F on a normed vector space \mathcal{X} is a γ -contraction, for $0 < \gamma < 1$, provided for all $x, y \in \mathcal{X}$

$$\|F(x) - F(y)\| \leq \gamma \|x - y\| \quad (7)$$

Theorem 7.2

Contraction Mapping Theorem

For a γ -contraction F in a complete normed vector space \mathcal{X} , F converges to a unique fixed point in \mathcal{X}

이제, policy evaluation 알고리즘의 수렴성을 보이겠습니다. 먼저 Contraction Mapping Theorem을 보이기 위해서는 γ -contraction mapping을 찾아야 합니다. 여기서 5를 정리하면 다음과 같습니다.

$$\begin{aligned} v_\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} T_{s,s'}^a v_\pi(s') \right) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a|s) T_{s,s'}^a v_\pi(s') \end{aligned} \quad (8)$$

Convergence of policy evaluation-cont.

이 때, $r_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a$, $T_{s,s'}^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) T_{s,s'}^a$ 로 정의한 뒤, $v_\pi(s)$ 를 열벡터로 만들어 v_π 로 표기하면 5은 다음과 같은 행렬 방정식으로 정리됩니다.

$$v_\pi = r^\pi + \gamma T^\pi v_\pi \quad (9)$$

이 때, $F^\pi(v) = r^\pi + \gamma T^\pi v$ 로 정의하면 다음과 같은 관계를 만족합니다.

$$\begin{aligned} \|F^\pi(u) - F^\pi(v)\|_\infty &= \|(r^\pi + \gamma T^\pi u) - (r^\pi + \gamma T^\pi v)\|_\infty \\ &= \|\gamma T^\pi(u - v)\|_\infty \\ &\leq \gamma \|T^\pi\|_\infty \|u - v\|_\infty && \text{(Property of matrix } \infty\text{-norm)} \\ &= \gamma \|u - v\|_\infty && \text{(Matrix } \infty\text{-norm: largest row sum)} \end{aligned} \quad (10)$$

따라서 $F^\pi(v)$ 가 γ -contraction mapping임을 증명하였고, 이를 통해 $F^\pi(v)$ 는 유일한 고정점인 실제 v_π 로 수렴함을 알 수 있습니다. \square