Experience-driven Networking: A Deep Reinforcement Learning based Approach

INFOCOM 2018

Background

1.Traffic Engineering(TE) Problem
A set of network flows with source and destination nodes
A solution to forward data traffic with one objective function

Existing solutions:
1.Open shortest path first(OSPF)
2.Valiant load balancing(VLB)
both are not very ideal solutions.

Problem Statement

a communication network: K end-to-end communication sessions

communication session k has a source node s_k, destination d_k and a set of candidate paths $\mathbf{P}_k$ (connecting s_k with d_k)

study a TE problem seeking a rate allocation solution, which specifies the amount of traffic load $f_{k,j}$ going through the jth path of $\mathbf{P}_k$.

Path j is chosen with probability:

$$w_{k,j} = f_{k,j}/(\sum_{j=1}^{|P_k|} f_{k,j}),$$

Problem Statement

Utility function: The utility of a communication session:

$$U_\alpha(x) = (\frac{x^{1-\alpha}}{1-\alpha}).$$

The objective of a TE problem:

$$\sum_{k=1}^{K} U_\alpha(x).$$

In order to address both throughput and delay:

Redefine the utility function:

$$U(x_k, z_k) = U_{\alpha_1}(x_k) - \sigma \cdot U_{\alpha_2}(z_k),$$

Problem Statement

Network Utility Maximization (NUM)

$$\max_{<x_k, f_{k,j}>} \sum_k U_\alpha(x_k) \tag{6a}$$

subject to:

$$\sum_{k=1}^{K} \sum_{\mathbf{p}_j \in \mathbf{P}_k : e \in \mathbf{p}} f_{k,j} \leq C_e, \forall\ e \in \mathbf{E}; \tag{6b}$$

$$x_k \leq B_k,\ k \in \{1, \cdots, K\}; \tag{6c}$$

$$\sum_{j=1}^{|\mathbf{P}_k|} f_{k,j} = x_k,\ k \in \{1, \cdots, K\}. \tag{6d}$$

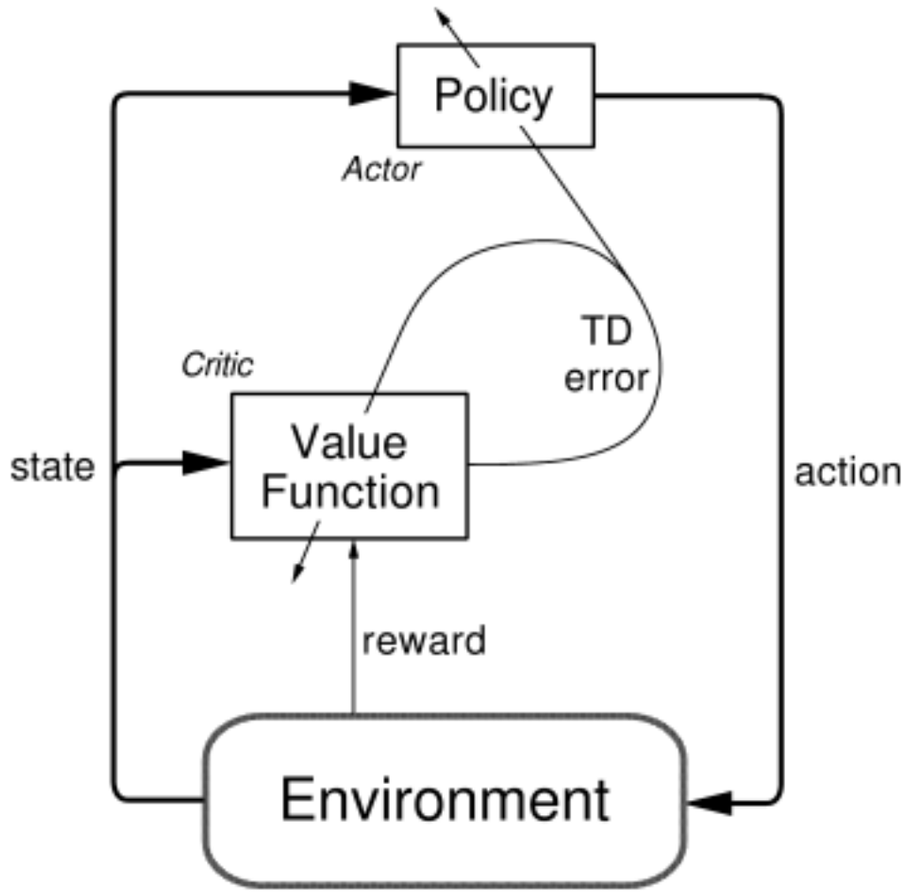
Background

DRL:
1.model-free
2.able to deal with highly dynamic time-variant environments
3.able to handle sophisticated state space

DQN: works on limited action space
DDPG: Deep Deterministic Policy Gradient

Background

DDPG:continuous action domain



Actor-critic model

Background

DDPG:continuous action domain

To stabilize, it also uses target network and experience replay.

Update for critic:

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E}\left[\left(Q(s_t, a_t|\theta^Q) - y_t\right)^2\right]$$

Update for actor:

$$\boldsymbol{\theta}^{\mu}_{k+1} = \boldsymbol{\theta}^{\mu}_{k} + \boldsymbol{\alpha}\mathbb{E}_{\mu'k}\left[\boldsymbol{\nabla}_{\theta} \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{\mu}(\boldsymbol{s}|\boldsymbol{\theta}^{\mu}_{k})|\boldsymbol{\theta}^{Q}_{k})\right].$$

by applying chain rule:

$$\boldsymbol{\theta}^{\mu}_{k+1} = \boldsymbol{\theta}^{\mu}_{k} + \boldsymbol{\alpha}\mathbb{E}_{\mu'k}\left[\boldsymbol{\nabla}_{a} \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}|\boldsymbol{\theta}^{Q}_{k})|_{a=\mu(s|\theta^{\mu}_{k})} \boldsymbol{\nabla}_{\theta} \boldsymbol{\mu}(\boldsymbol{s}|\boldsymbol{\theta}^{\mu}_{k})\right].$$

$$\begin{aligned}
\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)}\right] \\
&= \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t}\right]
\end{aligned}$$

Background

DDPG:continuous action domain

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Problem Statement

a communication network: K end-to-end communication sessions

communication session k has a source node s_k, destination d_k and a set of candidate paths $\mathbf{P}_k$ (connecting s_k with d_k)

study a TE problem seeking a rate allocation solution, which specifies the amount of traffic load $f_{k,j}$ going through the jth path of $\mathbf{P}_k$.

Path j is chosen with probability:

$$w_{k,j} = f_{k,j}/(\sum_{j=1}^{|P_k|} f_{k,j}),$$

DRL Framework

State Space: throughput and delay of each communication session to capture network states

$$\mathbf{s} = [(x_1, z_1), \cdots, (x_k, z_k), \cdots, (x_K, z_K)].$$

Action Space: Probability over all valid path for K sessions

$$[w_{1,1}, \cdots, w_{kj}, \cdots, w_{K,|P_k|}], \text{ where } \sum_{j=1}^{|P_k|} w_{k,j} = 1.$$

Reward: The objective function of TE problem

$$r = \sum_{k=1}^{K} U(x_k, z_k).$$

DRL Framework

Original DDPG does not perform well on this TE problem:
(1)DDPG does not clearly specify how to explore.

ε-greedy Exploration

$$\pi(s) = \begin{cases} \text{random action from } \mathcal{A}(s) & \text{if } \xi < \varepsilon \\ \text{argmax}_{a \in \mathcal{A}(s)} Q(s,a) & \text{otherwise,} \end{cases}$$

Only work for tasks with limited discrete action space

The author also tested the exploration method(adding noise to actions) in original DDPG algorithms, also does not work.

(2)DDPG just uses uniform random sampling for experience replay.

DRL Framework

TE-aware exploration:
Leverage a good TE solution as the baseline during exploration.

with ϵ probability,

action = $\mathbf{a}_{\text{base}} + \epsilon \cdot \mathcal{N}$;

Otherwise,

action = $\mathbf{a} + \epsilon \cdot \mathcal{N}$;

$\mathcal{N}$ is a uniformly distributed random noise.

ϵ decreases with decision epoch t

DRL Framework

Prioritized experience replay:
When combined with DQN, it can achieve better performance.
main idea:assign priority for each transition

For actor-critic architecture:

priority has two part:
1.Temporal-Difference(TD) error. for critic network

$$\delta = y - Q(\mathbf{s}, \mathbf{a}),$$

2.The Q value gradient, for actor network

$$\nabla_{\mathbf{a}} Q = \nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})|_{\mathbf{s}=\mathbf{s}_i, \mathbf{a}=\pi(\mathbf{s}_i)}$$

$$p = \varphi \cdot (|\delta| + \xi) + (1 - \varphi) \cdot \overline{|\nabla_{\mathbf{a}} Q|},$$

Probability for sampling a transition:

$$P(i) = \frac{p_i^{\beta_0}}{\sum_j^{|\mathbf{B}|} p_j^{\beta_0}},$$

DRL Framework

for i = 1 **to** N **do**
Sample a transition $(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}_{i+1})$ from $\mathbf{B}$ where $i \sim P(i) := p_i^{\beta_0} / \sum_j p_j^{\beta_0}$;
Compute important-sampling weight: $\omega_i := (|\mathbf{B}| \cdot P(i))^{-\beta_1} / \max_j \omega_j$;
Compute target value for critic network: $Q(\cdot)$ $y_i := r_i + \gamma \cdot Q'(\mathbf{s}_{i+1}, \pi'(\mathbf{s}_{i+1}))$;
Compute TD-error: $\delta_i := y_i - Q(\mathbf{s}_i, \mathbf{a}_i)$;
Compute gradient: $\nabla_{\boldsymbol{\theta}^\pi} J_i :=$ $\nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})|_{\mathbf{s}=\mathbf{s}_i, \mathbf{a}=\pi(\mathbf{s}_i)} \cdot \nabla_{\boldsymbol{\theta}^\pi} \pi(\mathbf{s})|_{\mathbf{s}=\mathbf{s}_i}$;
Update the transition priority: $p_i := \varphi \cdot (|\delta_i| + \xi) + (1 - \varphi) \cdot \overline{|\nabla_{\mathbf{a}} Q|}$;
Accumulate weight-change for critic network: $Q(\cdot)$ $\Delta_{\boldsymbol{\theta}^Q} := \Delta_{\boldsymbol{\theta}^Q} + \omega_i \cdot \delta_i \cdot \nabla_{\boldsymbol{\theta}^Q} Q(\mathbf{s}_i, \mathbf{a}_i)$;
Accumulate weight-change for actor network: $\pi(\cdot)$ $\Delta_{\boldsymbol{\theta}^\pi} := \Delta_{\boldsymbol{\theta}^\pi} + \omega_i \cdot \nabla_{\boldsymbol{\theta}^\pi} J_i$;
end for

DRL Framework

Network Design:

Actor: 2-layer fully connected neural network
with 64 and 32 nodes for two layers respectively.
Softmax as activation function

Critic: 2-layer fully connected neural network
with 64 and 32 nodes for two layers respectively.
Leaky Relu for activation function

Value of hyper-parameters:

$$\xi := 0.01,\ \beta_0 := 0.6,\ \beta_1 := 0.4,\ \gamma := 0.99,\ \varphi := 0.6,$$
$$\eta^{\pi} := 0.001,\ \eta^{Q} := 0.01,\ \tau := 0.01 \text{ and } N = 64.$$

Evaluation

Three Network Topologies:
1.NSF Network
20 nodes and 64 links

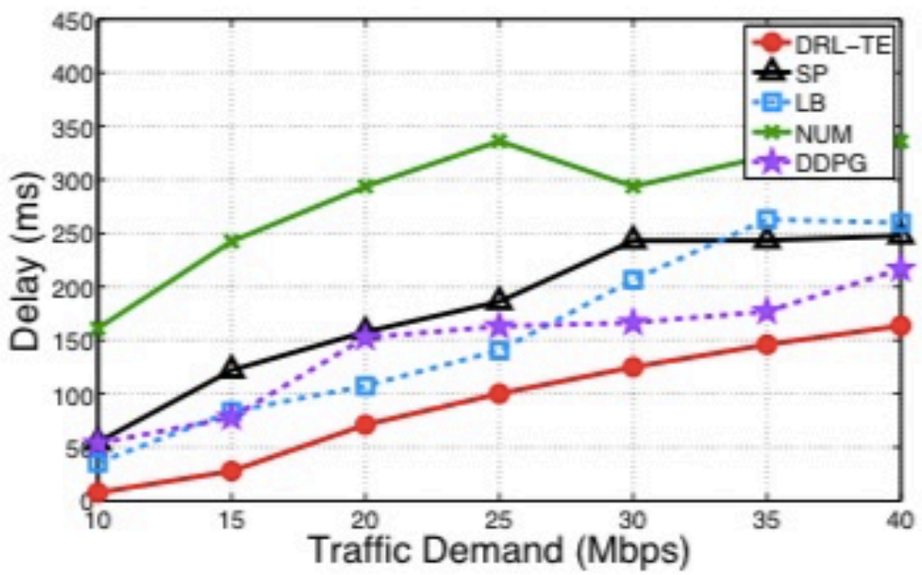
2.Advanced Research Projects Agency Network
14 nodes and 42 links

3.Randomly Generated Network via BRITE
20 nodes and 80 links

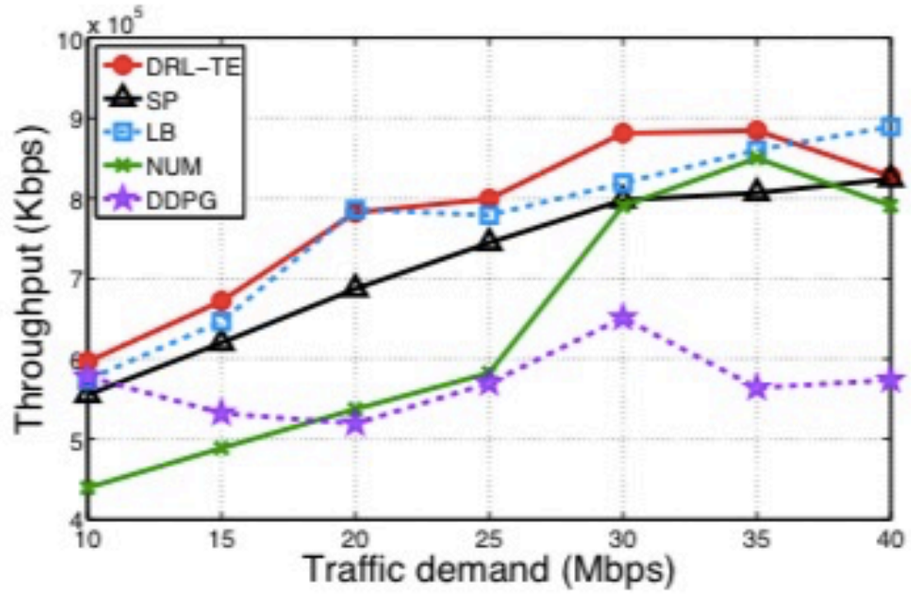
20 communication session
3-shortest paths as candidate path for each session

Evaluation

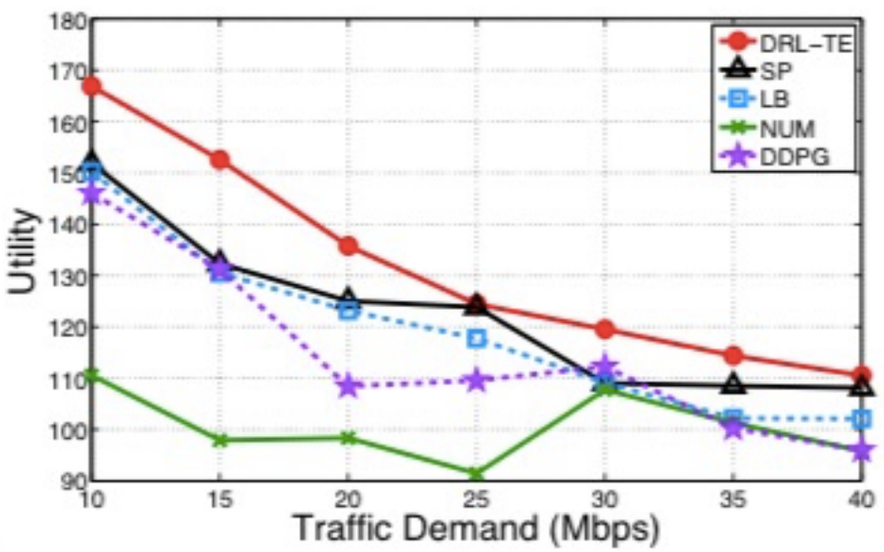
For NSFNET:



(a) End-to-end delay

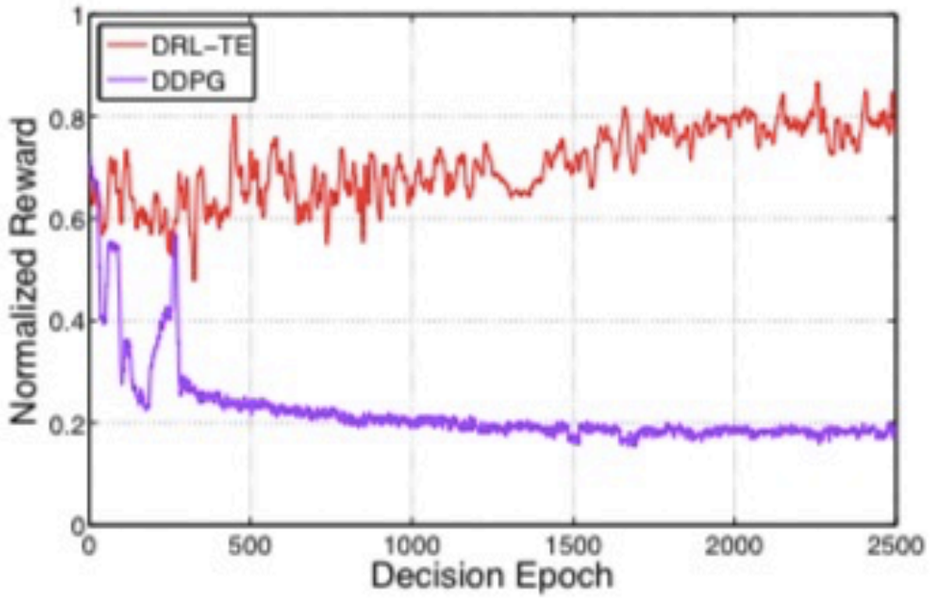


(b) End-to-end throughput

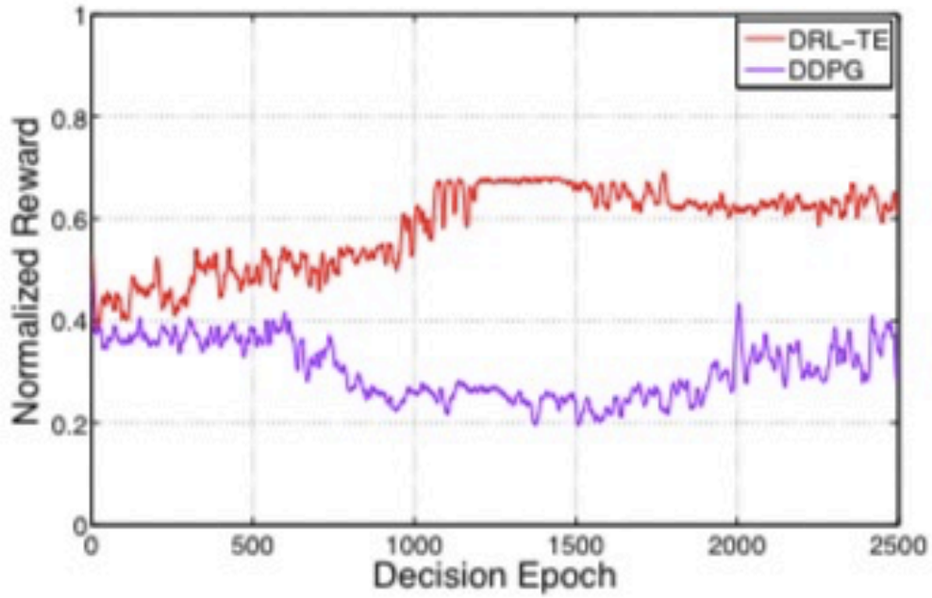


(c) Network utility

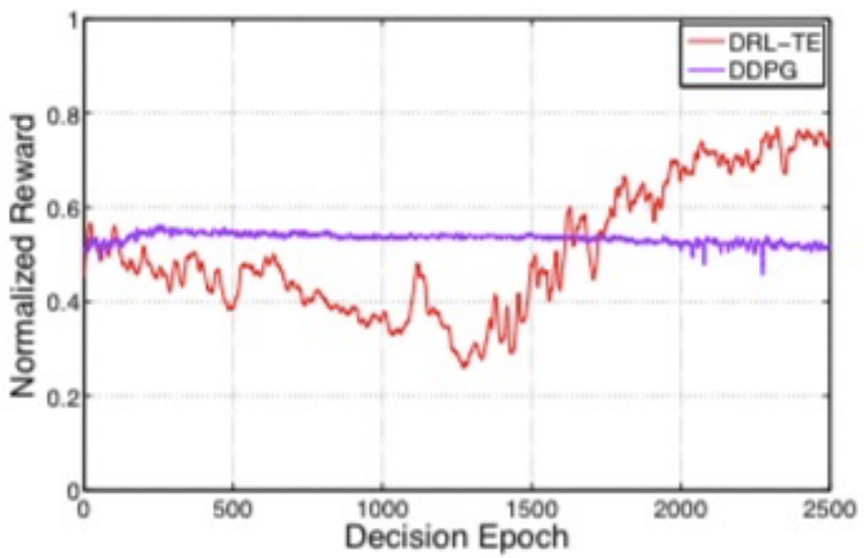
Evaluation



(a) NSFNET topology



(b) ARPANET topology



(c) Random topology

Conclusion

DRL is not as powerful as expected, we still need to adjust algorithm for specific setting to gain better performance. Exploration is critical for DRL, since it can provide some directions for DRL agent to learn, otherwise it can easily fall into local optimal.

DRL is sample-intensive, it needs to see as many samples as possible to gain experience about how to perform well. Every way to improve the performance of DRL is to make this learning process more efficient, that is to provide some directions to learn. For example, Prioritized replay and directed exploration.