Journal of KIIT. Vol. 22, No. 2, pp. 101-108, Feb. 28, 2024. pISSN 1598-8619, eISSN 2093-7571 101
http://dx.doi.org/10.14801/jkiit.2024.22.2.101

SDN에서 PEARL을 이용한 멀티캐스트 라우팅 트리 형성 기법

김희찬*, 김남기**

The Method for Creating a Multicast Routing Tree using PEARL in SDN

Heechan Kim*, Namgi Kim**

본 논문은 2024학년도 경기대학교 대학원 연구원장학생 장학금 지원에 의하여 수행되었음.
본 논문은 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임
(No.2020R1A6A1A03040583)

요 약

OTT 서비스(over-the-top media service)의 발전으로 인해 멀티미디어 트래픽의 양이 기하급수적으로 증가하고 있다. 멀티미디어 트래픽을 효율적으로 전송하기 위해서는 하나의 송신자에서 출발하여 여러 수신자까지의 경로를 결정하는 멀티캐스트 라우팅 트리를 효과적으로 설정하는 것이 매우 중요하다. 본 논문에서는 가변적인 네트워크 환경을 고려하여 메타 강화학습 모델 중 하나인 PEARL을 이용하여 효율적인 멀티캐스트 라우팅 트리 생성 방법을 제안한다. 메타 강화학습은 다양한 태스크를 학습할 수 있는 기법으로 가변적인 네트워크 환경에 적합하며 그 중 PEARL은 컨텍스트 기반 메타 강화학습 기법으로 기존 메타 강화학습에 비해 샘플 효율성이 높다. 가변적인 네트워크 환경에서 제안하는 방법의 성능을 측정하기 위해 링크 분포를 랜덤, 30%, 50%, 70%으로 바꾸며 실험하였다. 실험 결과, 제안 방법으로 학습한 모델의 성능이 1.184로 최적에 가까운 멀티캐스트 라우팅 트리를 생성할 수 있음을 확인했다.

Abstract

The amount of multimedia traffic is increasing exponentially due to the development of over-the-top media services(OTT). In order to efficiently transmit multimedia traffic, it is very important to effectively set up a multicast routing tree that determines the route from one sender to multiple receivers. In this paper, we propose an efficient multicast routing tree generation method using PEARL, one of the meta-RL models, considering the variable network environment. Meta-RL is a technique that can learn various tasks and is suitable for variable network environments, and PEARL is a context-based meta-RL technique with high sample efficiency compared to conventional meta-RL. To measure the performance of the proposed method in a variable network environment, we experimented with random, 30%, 50% and 70% link distributions. The experimental results show that the performance of the model trained by the proposed method is 1.184, which can generate a near-optimal multicast routing tree.

Keywords
network, SDN, deep learning, Meta-Reinforcement learning

* 경기대학교 컴퓨터과학과 석사과정
- ORCID: https://orcid.org/0009-0002-4930-3486
** 경기대학교 컴퓨터과학과 교수(교신저자)
- ORCID: https://orcid.org/0000-0002-0077-6576

· Received: Dec. 04, 2023, Revised: Jan. 03, 2024, Accepted: Jan. 06, 2024
· Corresponding Author: Namgi Kim
Divison of Computer Science and Engineering, Kyonggi University,
154-42, Gwanggyosan-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, Korea
Tel.: +82-31-249-9962, Email: ngkim@kgu.ac.kr

Ⅰ. 서 론

최근 실시간 스트리밍의 확대 및 5G의 보급에 더불어 4K, 8K 같은 고해상도 콘텐츠의 사용이 증가함에 따라 네트워크 트래픽의 양이 급속도로 증가하고 있다. 이에 효과적으로 멀티미디어 트래픽을 처리하기 위해 SDN(Software Defined Network)에 관한 연구가 많이 진행되고 있다. SDN은 기존 네트워크 환경에서 데이터 평면과 제어 평면을 분리하여 네트워크 제어를 중앙에서 효율적으로 제어할 수 있는 방법이다. 이러한 SDN 환경에서 하나의 송신자에서 출발하여 여러 수신자에게 멀티미디어 트래픽을 효율적으로 전송하기 위해서는 경로 비용이 최소화될 수 있는 최소 비용 멀티캐스트 트리를 생성해내는 기술이 반드시 필요하다. 이를 위해 최근 SDN 환경에 각광받고 있는 머신러닝 및 딥러닝 기술을 활용하고자 하는 연구들이 활발히 진행되고 있다[1]. 이러한 머신러닝 기술 중 최적 제어로부터 시작된 강화학습은 네트워크 관리와 최적화에 사용될 수 있다. 그런데 강화학습을 사용하여 SDN 환경에서 네트워크 최적화를 위한 여러 연구들이 많이 진행되었으나 기존 강화학습을 사용한 연구는 새로운 태스크에 대해선 성능이 떨어진다는 단점이 존재한다. 따라서 가변적인 네트워크 환경에서도 잘 적응하여 최적의 멀티캐스트 트리를 생성할 수 있는 방법이 필요하다. 본 논문에서는 이러한 문제를 해결하기 위해 기존 강화학습에서 여러 태스크를 학습하기 위해 고안된 기법인 메타 강화학습 기법을 활용하여 링크 가변적인 네트워크 환경에서 멀티캐스트 라우팅 트리를 효율적으로 만들기 위한 방법을 제안하고 그 성능을 평가한다.

Ⅱ. 관련 연구

2.1 SDN 환경에서의 멀티캐스트 라우팅 트리

SDN 환경에서 라우팅 트리를 생성하는 방법으로는 스타이너 트리 문제(Steiner tree problem)를 해결하는 것과 동일하다[2][3]. 스타이너 트리 문제의 시간 복잡도는 NP-Hard이므로 다항 시간에 해결할 수 없어 TM[4]와 같은 근사 알고리즘을 이용한 다양한 연구들이 존재한다[5]. 하지만 근사 알고리즘을 이용한 방법은 계산시간이 오래 걸린다는 단점이 있다. 반면, 딥러닝을 이용한 방법은 모델을 학습하는데 시간이 오래 걸릴 수 있다는 단점이 있지만 학습 이후에는 짧은 시간 내 결과를 도출한다는 장점이 있다. 따라서 본 논문에서는 메타 강화학습을 이용한 딥러닝 기반의 모델을 제안한다.

2.2 강화학습

강화학습은 인공지능에서 매우 중요한 주제 중 하나로 그림 1과 같이 학습하는 주체인 에이전트가 주어진 환경과 상호작용을 하면서 특정 목표를 달성한다. 이때 환경에 놓여진 에이전트가 액션을 수행하면 환경은 에이전트에 액션에 대응하여 다음 상태와 보상치를 부여한다. 이와 같이 강화학습은 최적의 행동을 선택하는 방법을 스스로 학습하며 보상을 통해 학습을 이루어 낸다. 하지만 이러한 강화학습 구조는 하나의 태스크에 대해서만 학습하여 단일 태스크에 대해서만 최적이므로 만일 학습한 태스크와 조금이라도 다른 태스크가 환경으로 주어진다면 최적의 행동을 선택하지 못한다. 따라서 이러한 강화학습 방법을 통해 학습하는 방식은 가변적인 네트워크 환경에 적합하지 않다. 이에 본 논문에서는 후술하겠지만 다양한 태스크가 환경으로 주어져도 최적의 행동을 선택할 수 있는 강화학습에 메타 러닝을 적용한 메타 강화학습을 이용하여 가변적인 네트워크 환경에서도 최적의 행동을 선택할 수 있는 효율적인 방법을 제안한다.

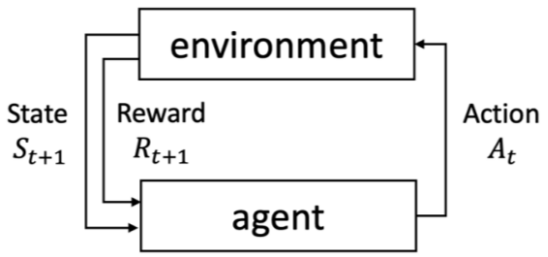

그림 1. 강화학습 구조
Fig. 1. Reinforcement learning structure

메타 강화학습은 크게 3가지 접근방법에 따라 나뉜다. 첫 번째는 순환 정책 메타 강화학습, 두 번째는 최적화 기반 메타 강화학습, 그리고 세 번째로 컨텍스트 기반 메타 강화학습이다. 식 (1)과 같이 메타 강화학습은 기존 강화학습과 달리 내부 단계에서 각각의 태스크인 MDP M_i에 대해 태스크 적응 $\Phi_i = f_\theta(M_i)$을 수행하며 외부 단계에서는 이를 통합하여 각 태스크의 보상합에 대한 기댓값을 최대화하는 최적의 메타 파라미터 θ^*을 찾는다.

$$\theta^* = argmax_\theta \sum_i^n E_{\pi_{\Phi_i(\tau)}}[R(\tau)], \quad where\, \Phi_i = f_\theta(M_i) \tag{1}$$

이러한 메타 강화학습은 동일한 태스크의 다양한 변종들이 존재하거나, 새로운 태스크가 주기적으로 등장하는 경우 유용하게 적용될 수 있다. 이는 유사하지만 링크 분포나 노드의 개수에 차이가 있는 가변적인 네트워크 환경에 적합하다. 본 논문은 메타 강화학습 모델 중 컨텍스트 기반 메타 강화학습 중 하나인 PEARL을 이용하여 고정되어 있지 않는 새로운 태스크인 링크 가변적인 네트워크 환경이 주어져도 최적에 가까운 멀티캐스트 라우팅 트리 생성 방법을 제안한다.

2.3 PEARL

PEARL[6]은 앞서 소개한 메타 강화학습 기법 중 컨텍스트 기반 메타 강화학습 알고리즘이다. 여기서 컨텍스트란 태스크를 유추할 수 있는 일련의 상호작용 데이터를 의미하며 M_i을 경험하며 추론된다. 에이전트는 이러한 컨텍스트 정보를 활용하여 메타 정책을 업데이트하고 새로운 태스크에서의 최적 행동을 선택한다. 따라서 식 (2)[6]처럼 컨텍스트 c로부터 추론된 잠재 컨텍스트 변수 z가 정책함수의 조건부에 붙게 된다.

$$\pi_\theta = (a|s,z),\ z \sim p(z|c) \tag{2}$$

컨텍스트 기반 메타 강화학습의 대표적인 모델인 PEARL은 순환 정책 메타 강화학습의 대표적인 모델인 RL^2[7] 와 최적화 기반 메타 강화학습의 대표적인 모델인 MAML[8]이 on-policy 기반이기 때문에 샘플 효율성이 낮고 새로운 태스크를 학습할 시 새로운 태스크의 불확실성을 구조적으로 탐색할 수 방법이 부족하다는 단점을 해결하였다. PEARL은 off-policy 기반의 메타 강화학습 알고리즘으로 Replay Buffer를 활용하여 높은 샘플 효율성을 보이며 태스크 추론을 위해 강화학습 부분과 태스크 추론 부분을 분리한 방법을 제안하였다. PEARL에서 각 태스크는 MDP로 표현하되, 상태 전이 확률과 보상함수는 알려지지 않았으므로 식 (3)[6]와 같이 태스크를 정의한다. 또한 식 (4)[6]와 같이 컨텍스트를 정의하여 수집 및 샘플링한 N개의 컨텍스트를 $C_{1:N}^T$ (이하 c)과 같이 정의한다.

$$T = \{p(s_0), p(s_{t+1}|s_t,a_t), r(s_t,a_t)\} \tag{3}$$

$$c_n^T = (s_n, a_n, r_n, s'_n) \tag{4}$$

PEARL에서는 현재 태스크를 어떻게 수행해야하는 지에 대한 지식을 확률적 잠재 컨텍스트 변수 z로 묘사하며 정책을 $\pi_\theta(averts,z)$로 조건화하여 태스크에 대한 행동을 적응시킨다. 이러한 확률적 잠재 컨텍스트 변수인 z를 잘 추론하기 위해선 컨텍스트 c가 주어졌을 때 posterior $p(zvertc)$를 추론해야 한다. 하지만 이 posterior $p(zvertc)$를 계산하는 것은 어렵기 때문에 Φ를 매개변수로 하는 추론 네트워크 $q_\Phi(zvertc)$를 학습하여 posterior $p(zvertc)$를 근사한다. 근사하기 위해 PEARL은 생성 모델 VAE[9]의 변분적 추론 개념을 바탕으로 추론 목적함수를 정의하였고 그 식은 식 (5)과 같다.

$$E_T\left[E_{z\sim q_\Phi(z|c^T)}\left[\begin{matrix}R(T,z) + \\ \beta D_{KL}\left(q_\Phi(z|c^T)\,\|\,p(z)\right)\end{matrix}\right]\right] \tag{5}$$

식 (5)[6]에서 E_T는 모든 태스크에 대해 기댓값을 취하며 $E_{z\sim q_\Phi(zvertc^T)}$는 추론 네트워크 $q_\Phi(zvertc^T)$를 이용해 컨텍스트 변수 z를 샘플링하여 이를 이용해 $R(T,z) + \beta D_{KL}\left(q_\Phi(zvertc^T)\,\|\,p(z)\right)$의 값을 계산한다. 여기서 $p(z)$는 z에 대한 unit Gaussian으로 가정한다.

$R(T,z)$는 식 (2)에서 보듯 에이전트인 $\pi_\theta=(averts,z)$을 학습함에 따라 z를 잘 추론했는지 판단하는 목적함수가 되며 $\beta D_{KL}\left(q_\Phi(zvertc^T)\|p(z)\right)$는 z와 c사이의 상호 정보량을 제한하는 정보 병목 현상[10]에 대한 변분적 근사(Variational approximation)로 해석한다.

PEARL에서는 추론 네트워크를 설계할 때 관련 없는 종속성을 모델링하지 않으면서 태스크와 관련된 정보에 대한 최소한의 정보를 포착할 수 있을 만큼 충분히 표현력 있도록 구성하였다. 여기서 완전히 관찰 가능한 MDP의 인코딩은 permutation invariant 해야 한다. 즉, PEARL은 MDP 모델을 알기 위해 태스크가 무엇인지 추론할 필요도 없고 가치함수를 학습하기 위해 순서를 고려할 필요없이 단지 $\{s_i,a_i,r_i,s'_i\}$와 같은 데이터만 있으면 충분히 학습할 수 있다. 이러한 관찰을 기반으로 PEARL에서는 식 (6)[6]과 같이 $q_\Phi(zvertc_{1:N})$를 permutation invariant 표현을 사용하여 모델링한다.

$$q_\Phi(z|c_{1:N}) \propto \prod_{n=1}^{N} \Psi_\Phi(z|c_n) \tag{6}$$

여기서 $\Psi_\Phi(zvertc_n)$는 정규분포에서 나오는 값으로 모델링 할 수 있으며 쉽게 다루기 위해, 가우시안인자 $\Psi_\Phi(zvertc_n)=N\left(f_\Phi^\mu(c_n),f_\Phi^\sigma(c_n)\right)$로 표현할 수 있고 이로 인해 가우시안 사후 분포가 생성된다. PEARL은 이 값들을 1 부터 N 까지 다 곱하여 $q_\Phi(zvertc_{1:N})$를 모델링할 수 있다. 여기서 함수 f_Φ는 c_n에 대한 함수로 Φ로 매개변수화 된 신경망으로 표현되며 평균 μ와 표준편차 σ을 출력한다.

위와 같이 PEARL은 확률적 잠재 컨텍스트를 모델링함에 따라 posterior 샘플링이 가능하며 이를 통해 탐험을 효율적으로 할 수 있다는 장점을 가진다. 그 결과 PEARL은 충분히 태스크 정보를 학습했을 때 테스트 과정에서 접하는 새로운 태스크에 대해 posterior 샘플링을 통해 빠르게 태스크 정보를 추론하고 이를 학습할 수 있게 된다.

III. 제안하는 방법

3.1 SDN 환경에 대한 MDP 정의

본 논문은 PEARL을 기반으로 SDN 환경에서 최적의 멀티캐스트 라우팅 트리를 생성하는 방법에 대해 제안한다. 따라서 최적의 멀티캐스트 라우팅 트리를 만들기 위해서 최소 비용의 트리를 생성한다. 이것은 출발지에서부터 목적지까지 순서에 따라 어떤 링크를 선택해야 하는지 결정하는 순차적 행동 결정 문제로 정의할 수 있고, 이러한 순차적 행동 결정 문제는 MDP(Markov Decision Process)와 같이 수학적으로 모델링할 수 있다. 본 논문에서는 표 1과 같이 태스크 별 MDP를 정의하였다.

표 1. 본 논문의 환경에 대한 MDP 정의
Table 1. MDP definition for the environment in this paper

Attribute	Description
State	The currently connected multicast routing tree Source node information Destination node information
Action	The set of reachable links in the currently connected multicast routing tree
Reward	Give +1 reward if the destination node is reached Give +0 reward if not at the destination node
Gamma	Discount factor

그림 2는 예시로 만든 노드 6개의 네트워크 토폴로지 그림이다.

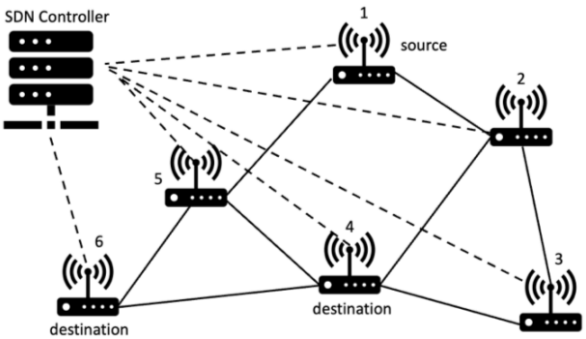

그림 2. 네트워크 토폴로지 도식도
Fig. 2. Network topology schematic

그림 3은 그림 2의 네트워크 토폴로지에 대한 인접행렬, 즉 상태 정보를 행렬로 나타낸 그림이다. 만일 노드 개수가 N 개면 $N \times N$인 인접행렬로 표현되며 이 인접행렬에는 출발지 노드 정보와 목적지 노드 정보가 같이 들어있다. 이 예시에서는 노드 개수가 6개이기 때문에 6×6인 인접행렬로 표현되었고 노드끼리 연결된 링크 정보를 나타내기 위해 현재 연결된 노드들 간 행렬값은 1로 표기하고 연결되지 않은 노드들 간 행렬값은 0으로 표기하였다. 즉 활성화된 링크는 1의 값을 가지며, 활성화되지 않는 링크는 0의 값을 가진다. 그리고 출발지 노드 정보와 목적지 노드 정보를 표기하기 위해 출발지 노드는 1로 목적지 노드는 -1로 표기하였다. 위 MDP에서 행동의 경우에는 현재 연결된 멀티캐스트 라우팅 트리에서 연결 가능한 링크 중 하나를 선택한다. 즉 활성화되어 있지 않은 링크를 선택한다. 마지막으로, 보상의 경우는 에이전트의 액션 이후, 즉 링크 선택 후 그 링크를 통해 목적지 노드 도달 여부에 따라 다르게 부여된다. 목적지 노드에 도달하였을 경우 +1, 도달하지 않았을 경우 +0을 부여한다. 또한 이 보상은 감쇠율이 적용되어 있어 여러 번 곱해질수록 0에 가까워지게 된다. 그러므로 에이전트는 누적 보상의 합을 최대로 하기 위해 최대한 적은 행동으로 에피소드를 끝내도록 학습될 것이다. 최대한 적은 행동을 통해 에피소드를 끝내는 것은 결국 최대한 적은 링크 개수로 멀티캐스트 라우팅 트리를 만드는 것이므로 이는 에이전트가 학습을 통해 최적의 멀티캐스트 라우팅 트리를 생성할 수 있음을 의미한다.

	1	2	3	4	5	6
1	1	1	0	0	1	0
2	1	0	1	1	0	0
3	0	1	0	1	0	0
4	0	1	1	-1	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	-1

그림 3. 그림 2 네트워크 토폴로지에 대한 인접행렬
Fig. 3. Adjacency matrix for the network topology in Fig. 2

3.2 PEARL을 이용한 제안 모델 및 학습 방법

PEARL의 메타 트레이닝 방법에 따라 본 논문에서 제안하는 모델 학습 방법은 다음과 같다. 앞서 언급하였듯 인코더 $q_{\Phi}(zvertc)$학습을 위해 사용되는 데이터가 정책을 학습하는 데 사용하는 데이터와 동일할 필요가 없다. 정책은 Off-policy 강화학습 과정에서 컨텍스트 z을 상태의 일부로 다룰 수 있으며, 탐험 과정의 확률적 요소는 인코더 $q_{\Phi}(zvertc)$의 불확실성에 의해 제공된다. Actor 네트워크와 Critic 네트워크는 항상 전체 리플레이 버퍼에서 샘플링한 Off-policy 데이터로 학습된다. 여기서 인코더 $q_{\Phi}(zvertc)$학습을 위해 컨텍스트 배치를 샘플링하기 위한 새로운 샘플러 S_c를 정의한다. 만일 이 S_c가 리플레이 버퍼와 동일하게 샘플링된다면 On-policy 데이터와 극단적인 분포 불일치를 보이게 될 것이다. 그러나 컨텍스트 배치는 완전히 On-policy일 필요는 없으며 최근 수집된 데이터의 버퍼에서 샘플링하여 인코더 $q_{\Phi}(zvertc)$를 학습하는 중간 전략을 택한다면 Off-policy의 장점을 가져가며 효율적으로 정책 성능을 유지할 수 있다. 또한 PEARL에서는 Off-policy 강화학습 알고리즘으로 Soft Actor-Critic[11]을 선택했다. Soft Actor-Critic은 강화학습 알고리즘 중 좋은 샘플 효율성을 보이며 엔트로피 개념을 활용해 알고리즘에 내포된 확률적 모델링 요소들은 PEARL의 인코더 학습 시 사용되는 확률적 잠재 컨텍스트와 잘 통합될 수 있다.

그림 4는 제안 모델의 학습 방법을 그림으로 나타낸 것이다. 제안 모델의 학습 방법은 먼저 인코더 $q_{\Phi}(zvertc^i)$에서 컨텍스트 변수인 z를 샘플링한다. 샘플링한 컨텍스트 변수 z를 통해 Actor network인 $\pi_{\theta}(averts,z)$에서 액션인 a를 샘플링하고 생성된 데이터인 (s,a,r,s')을 리플레이 버퍼 B^i에 저장한다. 이후 리플레이 버퍼 B^i에서 저장된 데이터인 (s,a,r,s')을 샘플링하여 컨텍스트 변수 c^i를 업데이트한다. 앞서 소개한 새로운 샘플러 S_c를 통해 인코더 $q_{\Phi}(zvertc^i)$를 업데이트하기 위해 컨텍스트 변수 c^i를 샘플링한다. 그리고 Soft Actor-Critic agent을 업데이트하기 위해 리플레이 버퍼 B^i에서

RL batch b^i를 샘플링한다. 그다음 인코더 $q_\Phi(zvertc^i)$에서 컨텍스트 변수 z를 샘플링한 다음 샘플링한 컨텍스트 변수 z와 RL batch b^i를 이용하여 Actor network의 손실함수인 L^i_{actor}와 Critic Network의 손실함수인 L^i_{critic}을 계산한다. 이후 앞서 소개한 변분적 추론으로 얻은 전체 목적 함수의 정규화 항인 $\beta D_{KL}(q(zvertc^i)\|r(z))$을 계산하여 정규화 손실함수인 L^i_{KL}을 계산한다. 이후 Critic Network의 손실함수 L^i_{critic}와 정규화 손실함수 L^i_{KL}을 이용하여 인코더 parameter Φ를 업데이트하고 Actor 손실함수 L^i_{actor}을 이용해 Actor parameter θ_π를 업데이트하고 Critic 손실함수 L^i_{critic}을 이용해 Critic parameter θ_Q를 업데이트한다.

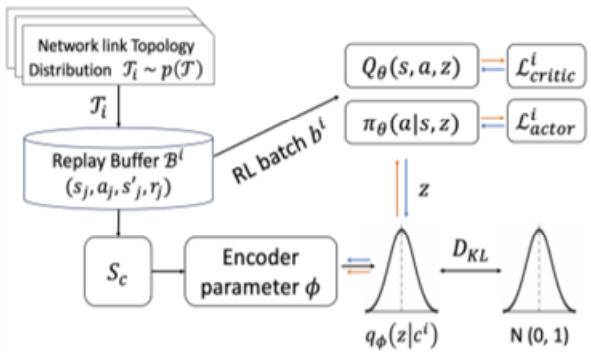

그림 4. 제안 모델 학습 방법
Fig. 4. How to train a suggestion model

IV. 실험 결과

4.1 성능 분석 기준

일반적인 강화학습에서는 누적 보상 또는 평균 보상에 따라 학습 정도를 파악한다. 그러나 본 논문에서는 태스크마다 목적지의 개수가 달라 얻을 수 있는 보상이 다르므로 누적 보상에 따라 성능을 평가하는 것은 적절하지 않다. 이에 본 논문에서는 각 태스크에서 만든 멀티캐스트 라우팅 트리 링크의 수를 활용하여 성능 평가 지표로 이용하였다.

$$Performance\ Ratio = \frac{Length\ of\ Multicast\ Routing\ Tree}{count(D_{destination})} \quad (7)$$

출발 노드와 목적 노드의 집합이 모두 연결되기 위해서는 목적 노드의 집합 원소 수만큼 링크의 개수가 필요하다. 따라서 필요한 최소 링크 개수는 목적지 집합 원소의 개수와 같다. 그다음 만들어진 멀티캐스트 라우팅 트리 링크의 개수에 최소 필요 노드 개수를 나누어 식 (7)과 같이 성능 평가 지표인 Performance Ratio를 정의한다. 식 (7)의 Performance Ratio는 1에 가까우면 가까울수록 최적의 멀티캐스트 라우팅 트리에 가까워진다고 볼 수 있다.

4.2 실험 결과

다음 실험은 가변적인 네트워크 환경을 고려하기 위해 네트워크 노드 개수가 10개인 환경에서 링크 분포를 변화해가며 iteration에 따른 Performance Ratio의 변화를 기록한 그래프이다. 실험은 링크 분포가 random일 때, 링크 분포가 30%일 때, 링크 분포가 50%일 때, 링크 분포가 70%일 때로 총 4회 진행하였다.

그림 5는 네트워크 노드 개수가 10개일 때 iteration이 지남에 따라 Performance Ratio를 기록한 그래프이다. 약 200 iteration 이후 대략 1.2 정도의 값에 수렴함을 보인다.

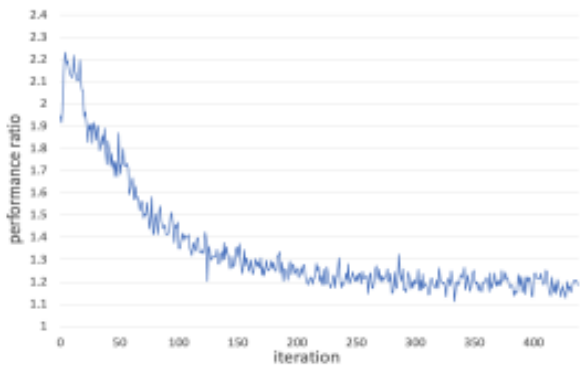

그림 5. 링크 분포가 랜덤일 때의 Performance Ratio
Fig. 5. Performance Ratio when link distribution is random

그림 6은 링크 분포가 30%일 때의 Performance Ratio이다. 약 300 iteration 이후 대략 1.5 정도의 값에 근접함을 알 수 있다.

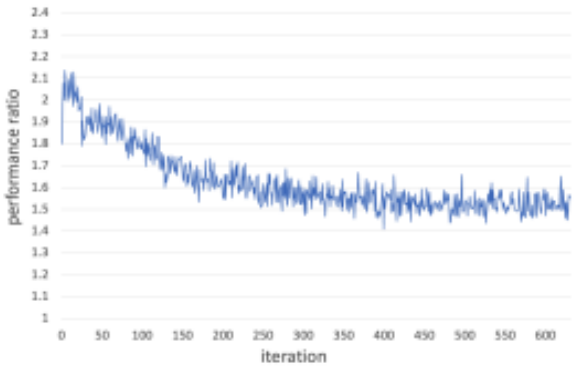

그림 6. 링크 분포가 30% 일 때의 Performance Ratio
Fig. 6. Performance Ratio when link distribution is 30%

그림 7은 링크 분포가 50% 일 때의 Performance Ratio이다. 약 300 iteration 이후 대략 1.2 정도의 값에 근접함을 알 수 있다.

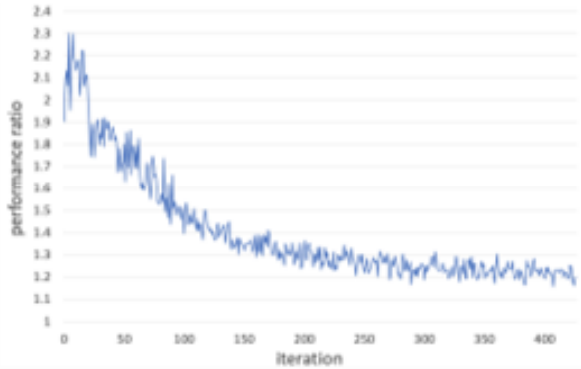

그림 7. 링크 분포가 50% 일 때의 Performance Ratio
Fig. 7. Performance Ratio when link distribution is 50%

그림 8은 링크 분포가 70%일 때의 Performance Ratio이다. 약 250 iteration 이후 대략 1.08 정도의 값에 근접함을 알 수 있다.

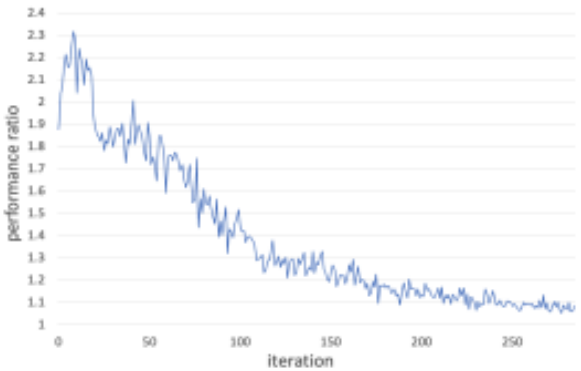

그림 8. 링크 분포가 70% 일 때의 Performance Ratio
Fig. 8. Performance Ratio when link distribution is 70%

그림 6, 그림 7, 그림 8의 실험 결과를 보면 링크 분포가 30% 일 때보다 링크 분포가 70%일 때 성능이 더 좋음을 알 수 있다. 이를 통해 링크 분포가 적을 경우 학습이 링크 분포가 많을 경우에 비해 어려움을 알 수 있다. 그 이유로는 링크 분포가 많을수록 탐색 공간과 경로가 많아 최적인 멀티캐스트 라우팅 트리를 만들 확률이 높고 어떤 방향으로 가든 최적에 가까울 확률이 높기 때문이다.

다음은 기존 근사 알고리즘[12]과의 비교이다. 표 2는 본 논문에서 제시한 성능 평가 지표인 Performance Ratio에 따라 기존 근사 알고리즘과 본 논문 간의 성능을 분석한 표이다. 위 표에서 보이는 것처럼 기존 근사 알고리즘보다 본 논문에서 제시한 모델의 성능이 뛰어남을 알 수 있다.

표 2. Performance Ratio 비교[12]
Table 2. Compare Performance Ratio[12]

Performance Ratio	Authors
2.000	Takahashi, Matsuyama
1.834	Zelikovsky
1.734	Berman, Ramaiyen
1.694	Zelikovsky
1.667	Promel, Steger
1.644	Karpinski, Zelikovsky
1.598	Hougardy, Promel
1.184(-0.414)	Ours(random)
1.529(-0.069)	Ours(30%)
1.197(-0.401)	Ours(50%)
1.086(-0.512)	Ours(70%)

V. 결 론

본 논문에서는 링크가 가변적인 네트워크 환경에서 메타 강화학습 모델인 PEARL[6]을 이용해 멀티캐스트 라우팅 트리 형성 방법을 제안했다. 실험 결과 Performance Ratio가 약 1.18 정도의 값으로 성능이 뛰어났다. 따라서 근사 알고리즘을 사용하는 것보다 메타 강화학습 알고리즘을 사용한 딥러닝 기반의 알고리즘을 사용하는 것이 성능과 시간 면에서 매우 이점이 있다는 것을 알 수 있었다. 향후 연구로는 본 실험이 네트워크 환경의 노드 개수가 10개이기에 비교적 작은 네트워크 규모에서 실험을 진행한 한계가 존재하기 때문에 노드 개수가 10개 이상인 환경에서 연구를 진행할 계획이다.

또한 실제 네트워크 환경은 링크뿐만 아니라 노드까지 가변적으로 변한다. 따라서 링크 개수뿐만 아니라 노드 개수까지 변하는 환경에서 메타 강화학습 기법을 적용하는 연구를 진행해 볼 계획이다.

References

[1] C. Yu, J. Lan, Z. Guo, and Y. Hu, "DROM: Optimizing the Routing in Software-Defined Networks With Deep Reinforcement Learning", in IEEE Access, Vol. 6, pp. 64533-64539, Oct. 2018. https://doi.org/10.1109/ACCESS.2018.2877686.

[2] P. Winter, "Steiner problem in networks: a survey", Networks, Vol. 17, No 2, pp 129-167, 1987. https://doi.org/10.1002/net.3230170203.

[3] F. K. Hwang and D. S. Richards, "Steiner tree problems", Networks, Vol. 22, No. 1, pp. 55-89, Jan. 1992. https://doi.org/10.1002/net.3230220105.

[4] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs", Math. Japonica, Vol. 24, No. 6, pp. 573-577, 1980.

[5] A. Z. Zelikovsky, "An 11/6-approximation algorithm for the network Steiner problem", Algorithmica, Vol. 9, pp. 463-470, May 1993. https://doi.org/10.1007/BF01187035.

[6] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine, "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables", arXiv preprint, arXiv:1903.08254, Mar. 2019. https://doi.org/10.48550/arXiv.1903.08254.

[7] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, L. Sutskever, and P. Abbeel, "RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning", arXiv preprint, arXiv:1611.02779, Nov. 2016. https://doi.org/10.48550/arXiv.1611.02779.

[8] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks", arXiv preprint, arXiv:1703.03400, Mar. 2017. https://doi.org/10. 48550/arXiv.1703.03400.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", arXiv preprint, arXiv:1312.6114, Dec. 2013. https://doi.org/10.48550/arXiv.1312.6114.

[10] A. A. Alemi, L. Fischer, J. V. Dillon, and K. Murphy, "Deep variational informationa bottleneck", arXiv preprint, arXiv:1612.00410, Dec. 2016. https://doi.org/10.48550/arXiv.1612.00410.

[11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", arXiv preprint, arXiv:1801.01290, Jan. 2018. https://doi.org/10.48550/arXiv.1801.01290.

[12] S. Hougardy and H. J. Promel, "A 1.598 Approximation Algorithm for the Steiner Problem in Graphs", Proc. of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland, Jan. 1999. http://dx.doi.org/10.1145/314500.314599.

저자소개

김 희 찬 (Heechan Kim)

2023년 2월 : 경기대학교 AI 컴퓨터공학부(공학사)
2023년 3월 ~ 현재 : 경기대학교 컴퓨터과학과 석사과정
관심분야 : 강화학습, 메타러닝, 네트워크

김 남 기 (Namgi Kim)

1997년 2월 : 서강대학교 컴퓨터과학과(공학사)
2000년 3월 : KAIST 전산학과(공학석사)
2005년 3월 : KAIST 전산학과(공학박사)
2007년 2월 : 삼성전자 통신연구소 책임연구원
2007년 3월 ~ 현재 : 경기대학교 컴퓨터공학부 교수
관심분야 : 통신시스템, 네트워크