https://doi.org/10.7840/kics.2023.48.11.1418

강화학습 기반 다차원 배낭 문제 해결에 대한 일반화 성능 향상 접근법

최 요 한◆, 석 영 준*, 김 주 봉*, 한 연 희°

A Generalization Performance Improvement Approach for Reinforcement Learning-Based Multidimensional Knapsack Problem Solving

Yohan Choi◆, Yeong-Jun Seok*, Ju-Bong Kim*, Youn-Hee Han°

요 약

조합 최적화 문제 중 하나인 배낭 문제는 NP-hard 문제로서 다항 시간 내에 최적해를 구하는 방법이 알려지지 않은 문제이다. 이러한 배낭 문제에 대한 해결책은 물류 및 창고 관리, 제조 및 생산 계획, 자원 할당 및 스케줄링 등 여러 분야에서 활용될 수 있다. 최근 배낭 문제를 강화학습을 통해 해결하려는 시도가 있다. 하지만 여러 연구가 배낭 문제의 물건 개수에 종속적인 방법들을 제안하여 주어진 물건의 개수가 바뀔 때마다 개별적으로 모델을 학습해야 한다는 단점을 가졌다. 본 논문은 배낭 문제가 가지는 규모 불변 특성을 활용하여 물건의 개수와 무관한 마르코프 결정 과정과 신경망 구조를 제안한다. 결과적으로 배낭 문제를 확장한 문제인 다차원 배낭 문제를 물건의 개수와 관련 없이 학습 및 사용 가능한 방법을 제안하고 실험을 통해 성능을 테스트한다. 추가적으로 제안하는 방법의 강점인 일반화 성능을 테스트하여 해당 방법이 확장성, 재사용성, 일반성을 가지는 것을 보인다.

키워드 : 심층학습, 강화학습, 조합 최적화 문제, 배낭 문제, 어텐션 메커니즘
Key Words : Deep Learning, Reinforcement Learning, Combinatorial Optimization Problem, Knapsack Problem, Attention Mechanism

ABSTRACT

The Knapsack Problem, one of the combinatorial optimization problems, has no known method for finding the optimal solution within polynomial time. Potential applications of solutions to this problem include logistics and warehouse management, manufacturing and production planning, resource allocation, and scheduling. Recently, attempts have been made to solve the Knapsack Problem using reinforcement learning. However, many proposed approaches are dependent on the number of items in the problem, requiring individual model training for each set of items. This paper proposes a Markov Decision Process and a neural network structure

※ 이 논문은 2023년도 한국기술교육대학교 교수 교육연구진흥과제 지원에 의하여 연구이며, 또한 2023년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(No. NRF-2023R1A2C1003143)의 연구임.
◆ First Author : Future Convergence Engineering, Korea University of Technology and Education, yoweif@koreatech.ac.kr, 학생회원
° Corresponding Author : Future Convergence Engineering, Korea University of Technology and Education, yhhan@koreatech.ac.kr, 종신회원
* Future Convergence Engineering, Korea University of Technology and Education, dsb04163@koreatech.ac.kr, 학생회원; rlawnqhd@koreatech.ac.kr
논문번호 : 202307-009-C-RN, Received July 10, 2023; Revised August 31, 2023; Accepted August 31, 2023

that exploits the scale-invariance of the Knapsack Problem and is independent of the number of items. As a result, the method proposes and tests the performance of an approach applicable to the extended Multi-dimensional Knapsack Problem, independent of the number of items. The method's generalization performance, a key strength, is evaluated to demonstrate its scalability, reusability, and generality.

Ⅰ. 서 론

최근 강화학습(Reinforcement Learning, RL)은 심층학습(Deep Learning)과 결합하여 인공지능 기술의 발전에 크게 이바지하고 있다. 강화학습은 인공지능의 한 분야로 에이전트(Agent)와 환경(Environment)의 상호작용에 따른 보상(Reward)을 통해 학습하는 방식이다. 예를 들어, 게임에서 승리하면 보상을 주고, 패배하면 보상을 주지 않는다. 이러한 보상을 통해 에이전트는 승리하는 방법을 스스로 찾아낸다. 이러한 강화학습은 바둑 세계 챔피언을 이기고[1-3], 비디오 게임에서 인간보다 높은 점수를 기록하는 등 다양한 분야에서 좋은 성과를 보이고 있다[4-6]. 또한, 강화학습을 활용해 조합 최적화 문제(Combinatorial Optimization Problem, COP)를 해결하려는 시도도 진행되고 있다[7-9].

본 논문은 조합 최적화 문제 중 물류 및 창고 관리, 제조 및 생산 계획, 자원 할당 및 스케줄링과 관련 있는 문제인 배낭 문제(Knapsack Problem)를 강화학습을 통해 해결하는 방법을 제안한다. 배낭 문제는 각각 무게와 가치를 가지는 물건들과 정해진 무게 제한이 있는 배낭이 있을 때, 배낭의 무게 제한을 넘지 않고 배낭 안에 있는 물건의 가치가 최대가 되도록 배낭에 담을 물건의 조합을 찾는 문제이다. 배낭 문제는 설명에 대한 이해가 쉽기 때문에 해결 방법도 쉬울 것이라고 생각될 수 있다. 하지만 배낭 문제는 NP-hard 문제로 다항 시간 안에 최적해를 구하는 방법이 알려지지 않은 문제이다.

배낭 문제의 최적해를 구하는 정확한 알고리즘(Exact Algorithm)으로서 동적 프로그래밍(Dynamic Programming) 및 분기 한정법(Branch and Bound) 등이 존재하지만, 물건의 개수가 많아질수록 계산 시간이 급격히 증가한다. 따라서 대규모 문제를 해결해야 하거나, 문제를 실시간으로 처리해야 할 때, 또는 컴퓨팅 자원이 제한된 엣지 디바이스 환경 같은 상황에서는 정확한 알고리즘을 적용하기 어렵다.

정확한 알고리즘에 대한 대안으로 배낭 문제를 해결하는 여러 근사 알고리즘(Approximation Algorithm)들이 연구되어 왔으며, 이들은 정확한 해보다 빠른 시간 안에 최적해에 가까운 해를 찾는 것을 목표로 한다. 근사 알고리즘으로서 휴리스틱(Heuristic) 및 메타휴리스틱(Metaheuristic) 알고리즘[10]들이 제안되었고, 이러한 방법들은 합리적인 시간 내에 만족스럽거나 최적에 가까운 해답을 얻는다는 점에서 장점이 있다. 하지만, 문제를 구성하는 조건이 약간만 변경되더라도 알고리즘을 수정해야 한다는 단점이 있다[7].

최근 심층학습의 발전과 함께 조합 최적화 문제를 해결하기 위해서 심층학습을 이용하는 시도가 학계에서 보고되고 있다. 심층학습은 이전까지의 휴리스틱 및 메타휴리스틱 알고리즘과는 다르게, 문제에 대한 조건이 다소 변경되더라도 알고리즘을 수정할 필요 없이, 심층 신경망(Deep Neural Network)을 통해 스스로 해결책을 찾을 수 있다. 심층학습은 고차원의 복잡한 데이터 표현(Representation)에 대한 학습 능력이 뛰어나다. 따라서, 심층학습은 응용 가능성이 높아 문제의 복잡성이나 크기와 관계없이 문제의 변화에 쉽게 대응하고 확장하는 것을 기대할 수 있다.

Gu et al. (2018)는 지도학습과 심층학습을 결합하여 배낭 문제를 해결하는 방법을 제안하였다[11]. 그러나, 지도학습의 특성상 학습을 하기 위해 정답 데이터가 필요하다. 배낭 문제의 정답 데이터는 최적해이며, 이는 정확한 알고리즘을 통해 구할 수 있다. 배낭 문제를 심층학습으로 해결하려는 주된 동기 중 하나는 정확한 알고리즘을 사용하여 최적해를 구하는 것이 어려운 경우에 활용하기 위함이다. 그러나 지도학습을 사용하는 방법은 정답 데이터를 얻기 위해 정확한 알고리즘이 필요하다는 모순이 발생한다. 그러므로 지도학습을 통해 배낭 문제를 해결하는 것은 실질적으로 한계가 있다.

심층학습을 활용하는 또 다른 방법으로는 강화학습과 접목한 심층강화학습(Deep Reinforcement Learning)이 있다. 강화학습은 지도학습과 달리 정답 데이터가 필요하지 않아 지도학습의 한계를 극복할 수 있다. 최근에 심층강화학습을 통해 배낭 문제를 해결하는 연구가 발표되고 있으며[12,13], 각각의 연구들은 다른 분야의 심층강화학습 연구들과 마찬가지로 마르코프 결정 과정(Markov Decision Process)의 설계를 제시하고 이에 적합한 구체적인 알고리즘을 제안한다.

심층학습을 활용한 알고리즘은 문제에 대한 조건이 변경되더라도 알고리즘을 수정할 필요 없이 학습을 통해 스스로 해결책을 찾을 수 있는 장점이 있지만, 그러한 장점을 제대로 발휘하기 위해서는 심층학습 모델 및 학습방법에 대한 고도화된 설계가 필요하다. 심층학습

에서 모델을 학습하는 것은 많은 시간과 자원이 필요하고, 모델의 크기 및 문제의 복잡성이 증가함에 따라 학습 시간이 급격히 증가한다.

배낭 문제에서 문제에 대한 조건이 변경되는 간단한 예시는 전체 물건의 개수가 바뀌는 경우이다. 이와 같은 일을 실제 문제에 대입해보면 물류 및 창고 관리 문제에서 물품의 개수가 바뀌거나 자원 할당 및 스케줄링 문제에서 작업의 수가 바뀌는 경우에 대응되며, 이러한 경우는 실제 업무 환경에서 자주 발생할 수 있다. 배낭 문제에서 전체 물건의 개수가 바뀌는 경우, 심층학습을 통해 학습된 모델은 바뀐 문제에 대해 다시 학습해야 한다는 불편함이 따른다.

심층학습을 활용하여 조합 최적화 문제를 해결하는 이전의 연구들은 주어진 문제에서 아이템 또는 노드의 수가 바뀌어도 확장성을 가지며 학습된 모델을 계속해서 사용할 수 있다[7-9]. 그러나 해당 연구들은 외판원 문제(Travelling Salesman Problem, TSP)나 차량 경로 문제(Vehicle Routing Problem, VRP)와 같은 그래프 문제를 해결하는 것에만 집중한다. 이와 같은 그래프 기반 조합 최적화 문제는 주어진 문제에 존재하는 노드들이 그래프를 구성하고, 노드 간 관계를 가지며, 문제의 해로서 노드들의 최적 시퀀스를 요구한다. 이에 반해 배낭 문제는 문제에 주어진 물건들 사이에 명시적인 관계를 가지지 않고, 문제의 해로서 주어진 물건 집합의 부분 집합을 요구한다. 따라서, 이전의 조합 최적화 문제에서 연구된 방법은 배낭 문제에 적합하지 않으며, 새로운 방법을 모색할 필요가 있다.

심층학습은 문제의 복잡성이나 크기 변화에 대응하는 확장성을 기대할 수 있지만, 이전까지의 배낭 문제를 해결하는 심층학습 연구는 신경망 구조의 한계 때문에 확장성이 부족하다. 물건의 개수가 바뀌면 신경망의 입/출력부 뉴런의 수도 바뀌어 학습된 모델을 재사용할 수 없고 새로 학습해야 한다. 본 논문은 심층강화학습과 어텐션 메커니즘(Attention Mechanism)[14]을 활용하여, 배낭 문제에서 물건의 개수가 바뀌어도 모델을 다시 학습할 필요 없이 재사용할 수 있는 방법을 제안한다.

Ⅱ. 관련 연구

기계학습으로 조합 최적화 문제 또는 배낭 문제를 해결하려 한 다양한 연구들이 존재하며, 이번 장에서는 이러한 연구들을 소개한다.

Gu et al. (2018)는 배낭 문제를 해결하기 위해 지도학습과 포인터 네트워크 (Pointer Network)[15]를 결합한 방식을 제안했다[11]. 이 방식에서는 하나의 배낭 문제를 풀기 위해 하나의 입력과 하나의 출력을 사용한다. 입력으로 물건의 속성을 포인터 네트워크에 순차적으로 전달하며, 출력으로 하나의 시퀀스를 통해 선택할 물건을 차례대로 결정한다. 포인터 네트워크는 순환 신경망(Recurrent Neural Network, RNN)을 기반으로 한 신경망이기 때문에 출력이 입력 시퀀스의 순서에 영향을 받게 된다. 따라서 물건에 대한 순서가 없는 배낭 문제에 포인터 네트워크를 활용하는 것은 적합하지 않다.

이 논문에서 10개, 100개, 500개, 1,000개의 물건이 있는 배낭 문제에 대하여 실험을 진행하였고, 물건의 개수에 따라 별도로 신경망을 학습했다. 인스턴스 수에 대해서는 문제마다 10,000개의 훈련 데이터를 생성하여 학습했고, 1,000개의 테스트 데이터를 생성하여 평가했다. 정답 데이터는 동적 프로그래밍을 통해 최적해를 구하여 사용했다. 결과적으로, 이 논문에서 제안하는 방법은 최적해에 비해 상당히 낮은 성능을 보인다.

Afshar et al. (2020)는 배낭 문제를 강화학습으로 해결하는 방법을 제안했다[12]. 이 논문에서는 배낭 문제를 순차적 의사 결정 문제(Sequential Decision Making Problem)로 모델링하고, 강화학습 알고리즘 중 하나인 A2C(Advantage Actor-Critic) 알고리즘[16]을 사용하여 학습한다. 이때, 상태 집계(State Aggregation) 접근 방식을 제안하여 상태 공간을 줄이고, 신경망의 입력으로 사용할 수 있도록 상태를 벡터화한다. 네트워크의 입력으로 사용되는 상태는 남은 물건의 무게와 가치, 배낭의 남은 용량, 남은 물건의 개수, 남은 물건의 가치의 총합, 남은 물건의 무게의 총합으로 구성되어 $2n'+4$ 차원의 벡터가 된다. 이때, n'은 배낭에 넣지 않고 남은 물건의 개수이다. 에이전트의 행동은 각 타임스텝마다 배낭에 넣을 물건 하나를 선택한다. 이러한 과정은 배낭에 더 이상 물건을 넣을 수 없거나, 물건이 남지 않을 때까지 반복한다.

이 논문에서 50개, 300개, 500개의 물건이 있는 배낭 문제를 무작위 생성하여 실험을 진행하였고, 물건의 개수에 따라 별도로 신경망을 학습했다. 각 학습에 사용된 인스턴스 수는 30,000×n개이고, 1,000개의 인스턴스로 성능을 측정했다. n은 전체 물건의 개수이다. 결과적으로, 탐욕적 알고리즘보다 높고 최적해에 근접한 성능을 보인다.

Yildiz와 Beytullah (2022)도 마찬자지로 강화학습을 이용해 배낭 문제를 해결하는 방법을 제안했다[13]. 알고리즘으로 DQN을 사용했는데, 주목해야 할 점은 Q 신경망(Q-Network)을 완전 연결 신경망(Fully Connected Neural Network), 어텐션(Attention) 모델,

트랜스포머(Transformer) 모델로 각각 구성하여 세 신경망 구조의 성능을 비교했다는 점이다. 또한, 이들은 배낭 문제를 2차원 배낭 문제로 확장하여 제약 조건으로 무게뿐만 아니라 부피도 추가하였다. 이 논문의 특이한 점은 현재 상태에 모든 물건에 정보를 포함하는 대신, 단 하나의 물건에 대한 정보만 포함한다는 점이다. 에이전트의 행동은 현재 관찰된 물건을 배낭에 넣을지 말지 결정하는 것이다. 이처럼 상태를 설정하면 상태 공간의 크기가 현저히 줄어들고, 문제에서 물건의 개수가 변해도 학습된 신경망을 여전히 사용할 수 있다. 그러나 배낭 문제에서는 배낭에 물건을 넣을 때, 다른 물건들의 정보를 고려해야 하므로, 이 상태 설정은 적합하지 않다. 물건의 개수가 변해도 신경망을 사용만 할 수 있을 뿐이고, 성능은 보장되지 않기 때문에 결국 학습을 다시 해야 한다.

실험에서의 문제 인스턴스 구성은 다음과 같다. 배낭의 무게 제한과 부피 제한이 1,000으로 고정되며, 물건의 무게와 가치는 0부터 100까지의 값에서 무작위 샘플링된다. 물건 개수는 30개에 대해서만 학습한다. 실험 결과는 세 가지 신경망 구조 모두 최적해에 비해 상당히 낮은 성능을 보이고, 완전 연결 신경망, 어텐션 모델, 트랜스포머 모델 순서로 성능이 미세하게 증가했다. 단, 이 논문에서 어텐션 모델을 활용하는 방법은 본 논문이 제안하는 어텐션의 활용 방법과는 전혀 다르다. 30개의 물건에 대해 학습한 모델을 20개, 40개의 물건에서 테스트했을 때, 성능이 떨어지는 것을 보여 일반화 성능을 전혀 보장할 수 없었다.

Ⅲ. 배경 지식

이 장에서는 본 논문이 해결하고자 하는 문제인 다차원 배낭 문제, 문제를 해결하 는 접근법인 강화학습, 강화학습 알고리즘인 DQN, 그리고 신경망 구조로 사용되는 어텐션 메커니즘에 대해 설명한다.

3.1 배낭 문제

무게와 가치를 지니는 여러 물건과 정해진 무게 제한이 있는 배낭이 있을 때, 배낭의 무게 제한을 넘지 않고 배낭 안에 있는 물건의 가치가 최대가 되도록 배낭에 담을 물건의 조합을 찾는 문제를 배낭 문제라고 한다. 물건들의 집합 $\mathbb{I}$와 배낭 $\mathbb{K}$가 주어졌을 때, 배낭 문제 $P^1=(\mathbb{I},\mathbb{K})$는 다음과 같이 정의할 수 있다. $\mathbb{I}$에는 총 $n(=|\mathbb{I}|)$개의 물건이 존재하고, 임의의 물건 $I^i\in\mathbb{I}$은 해당 물건의 가치 v^i와 무게 w^i를 속성으로 가지고, 2차원 벡터로 표현된다. 배낭 $\mathbb{K}$는 무게 제한 W을 가진다. 물건 I^i의 선택 여부를 $x^i\in\{0,1\}$로 표현할 때, $\mathbb{I}^{sel}\subseteq\mathbb{I}$는 선택된 물건들의 집합으로 $\mathbb{I}^{sel}=\{I^i\in\mathbb{I}\,|\,x^i=1\}$이다. 문제의 인스턴스, 목적, 제약 조건을 수식화하여 표현하면 다음과 같다.

$$
\begin{aligned}
&P^1=(\mathbb{I},\mathbb{K})\\
&I^i=(v^i,w^i)\in\mathbb{I}\\
&\mathbb{K}=(W)\\
&\text{maximize}_{\mathbb{I}^{sel}}\sum_{I^i\in\mathbb{I}^{sel}}v^i\\
&\text{subject to}\sum_{I^i\in\mathbb{I}^{sel}}w^i\le W,\ \ \mathbb{I}^{sel}\subseteq\mathbb{I}
\end{aligned}
\tag{1}
$$

배낭 문제는 NP-hard 문제에 속하며, 다항 시간 내에 최적해를 구하는 방법이 밝혀지지 않았다. 현재까지 알려진 해결책은 정확한 알고리즘으로 동적 프로그래밍 및 분기 한정법이 있고, 근사 알고리즘으로 메타휴리스틱으로 유전 알고리즘(Genetic Algorithm)[17] 및 타부 서치(Tabu Search)[18] 등이 존재한다[10]. 최근에는 2장에서 설명한 대로 심층학습을 이용한 알고리즘도 제안되었다[11-13].

3.2 다차원 배낭 문제

배낭 문제의 제약 조건으로 물건의 무게만 고려하는 것이 아니라, 물건의 부피와 같이 다른 속성까지 고려하여 배낭 문제를 확장할 수 있다. 여러 개의 제약 조건을 고려하는 배낭 문제를 다차원 배낭 문제(Multi-dimensional Knapsack Problem, MKP)라고 한다. 본 논문은 다차원 배낭 문제를 해결하는 것을 목표로 한다. 임의의 물건 I^i가 가지는 m개의 제약 조건 속 성의 크기를 $r^{i1}, r^{i2}, \cdots, r^{im}$라 하고, 각 속성에 대한 배낭의 최대 용량을 $C^1, C^2, \cdots, C^m$라 할 때, 문제 인스턴스, 목적, 제약 조건을 다음과 같이 정의할 수 있다.

$$
\begin{aligned}
&P^m=(\mathbb{I},\mathbb{K})\\
&I^i=(v^i,r^{i1},r^{i2},\cdots,r^{im})\in\mathbb{I}\\
&\mathbb{K}=(C^1,C^2,\cdots,C^m)\\
&\text{maximize}_{\mathbb{I}^{sel}}\sum_{I^i\in\mathbb{I}^{sel}}v^i\\
&\text{subject to}\sum_{I^i\in\mathbb{I}^{sel}}r^{ij}\le C^j,\\
&\qquad\forall\, j\in\{1,2,\cdots,m\},\ \ \mathbb{I}^{sel}\subseteq\mathbb{I}
\end{aligned}
\tag{2}
$$

3.3 강화학습

강화학습(Reinforcement Learning, RL)은 기계학습의 한 분야로서, 에이전트(Agent)가 환경(Environment)과 상호작용하며 누적 보상(Cumulative Reward)을 최대화하는 최적의 전략 또는 정책(Policy)을 학습하는 것을 목표로 하는 방법이다. 강화학습의 핵심 요소로는 상태(State), 행동(Action), 보상(Reward), 정책(Policy) 등이 있다.

상태는 에이전트가 인식하는 주어진 환경의 모습을 나타내며, 이 정보를 바탕으로 에이전트는 행동을 결정한다. 이렇게 결정된 행동은 에이전트가 환경에 가하는 영향으로, 환경은 에이전트의 행동에 따라 변화한다. 에이전트는 환경과 상호작용을 통해 환경으로부터 상태 정보와 함께 보상을 얻게 된다. 이 보상은 에이전트의 행동에 따른 피드백으로 작용하며, 에이전트는 이러한 피드백을 통해 정책을 학습하게 된다.

3.4 마르코프 결정 과정

마르코프 결정 과정(Markov Decision Process, MDP)은 강화학습에서 사용되는 수학적 모델로, 순차적인 의사 결정 문제(Sequential Decision Problem)를 모델링하는 방법이다. 마르코프 결정 과정은 상태 집합, 행동 집합, 상태 전이 확률(Transition Probability) 함수, 보상 함수로 구성된다.

마르코프 결정 과정의 중요한 속성 중 하나는 마르코프 속성(Markov Property)이다. 이는 마르코프 결정 과정에서의 다음 상태와 보상이 오직 현재 상태와 행동에만 의존한다는 것을 의미한다. 이러한 마르코프 속성 덕분에 강화학습에서는 마르코프 결정 과정을 사용하여 복잡한 환경을 단순화하고 이해하기 쉬운 형태로 모델링할 수 있다. 따라서, 임의의 주어진 문제에 대해 강화학습을 적용할 때에는, 마르코프 속성을 만족하도록 마르코프 결정 과정의 상태 집합, 행동 집합, 상태 전이 확률 함수, 보상 함수를 주어진 문제에 맞추어 정의하는 것이 매우 중요하다.

Ⅳ. 제안하는 방법

본 논문은 기본적으로 배낭 문제의 규모 불변(Scale Invariance) 속성을 고려한다. 규모 불변이란 문제의 크기가 변해도 해당 문제의 기본 구조와 원리가 동일하게 유지되는 속성을 의미한다. 임의의 문제가 이러한 속성을 지닌다면, 해당 문제의 크기에 상관없이 동일한 방법론이나 해법을 사용할 수 있다. 예를 들어, 테트리스 혹은 직소 퍼즐을 해결하는 어떤 알고리즘이 있다면, 테트리스 보드의 크기 또는 직소 퍼즐의 크기가 달라져도 문제의 규모만 커질 뿐 문제의 기본 원리는 바뀌지 않기 때문에 동일한 알고리즘을 적용할 수 있다. 배낭 문제도 규모 불변 속성을 지닌 문제로써, 물건의 개수가 바뀌어도 문제의 기본 원리는 변하지 않고 문제를 푸는 데에 사용되는 알고리즘을 그대로 사용할 수 있다. 하지만, 심층학습을 통하여 배낭 문제를 해결하는 접근 방식은 신경망의 구조적인 한계로 인해 물건의 개수가 변경되면 신경망의 뉴런 수, 특히 입력 계층의 뉴런 수가 달라져 기존에 학습한 모델을 그대로 사용할 수 없는 문제가 있다. 본 논문은 이러한 한계를 극복하기 위해 적절한 MDP와 신경망 구조를 제안한다.

4.1 MDP 정의

3장의 식 (2)에서 다차원 배낭 문제의 정의를 제시했다. 이러한 다차원 배낭 문제를 강화학습으로 접근하기 위해서는 문제를 순차적 의사 결정 문제로 모델링한 뒤 MDP를 정의해야 한다. 다차원 배낭 문제는 문제와 해답에 순서가 없는 조합 최적화 문제이며, 임의의 문제 $P^m = (\mathbb{I}, \mathbb{K})$가 주어졌을 때, 해답 $\mathbb{I}^{sel} \subseteq \mathbb{I}$를 구하는 것이 목표이다. 다차원 배낭 문제를 순차적 의사 결정 문제로 모델링하면, 임의의 타임스텝 t에 대하여 문제 $P_t^m = (\mathbb{I}_t, \mathbb{K}_t)$가 주어졌을 때, 해당 타임스텝 t마다 배낭에 넣을 물건 $I^{sel} \in \mathbb{I}_t$를 선택하는 문제를 반복하여 푸는 것으로 볼 수 있다. 이때 타임스텝 t의 문제 P_t^m에서 임의의 물건을 선택하는 함수를 $f(P_t^m)$로 표기할 수 있다. 순차적 의사 결정 문제로 다루어지는 다차원 배낭 문제는 사실 타임스텝 t마다 새로운 문제를 푸는 것과 같다. 즉, 문제 $P_t^m = (\mathbb{I}_t, \mathbb{K}_t)$에서 임의의 물건을 선택한 후 다음 문제 $P_{t+1}^m = (\mathbb{I}_{t+1}, \mathbb{K}_{t+1})$를 푸는 것은 마치 완전히 새로운 문제 $Q_t^m = (\mathbb{I}_{t+1}, \mathbb{K}_{t+1})$를 푸는 것과 같다. 따라서 물건을 선택하는 함수 $f(\cdot)$는 물건의 개수와는 무관한 함수이고, 이전에 어떤 물건을 선택했는지와도 무관하다. 이를 통해 배낭 문제가 규모 불변성을 가지는 문제라는 것을 알 수 있다. 지금까지 배낭 문제를 순차적 의사 결정 문제로 모델링하였고, 이제 함수 $f(\cdot)$를 심층강화학습으로 근사시키기 위해 적절한 MDP을 설계한다.

4.1.1 행동

다차원 배낭 문제의 MDP 설계에서 행동은 함수 $f(P_t^m)$의 출력이 I^{sel}임에서 알 수 있듯이 $\mathbb{I}_t$에서 배낭

에 넣을 물건 하나를 선택하는 것이다. 유의해야 할 점은 행동 공간의 크기가 $|\mathbb{I}_t| = n_t$로 매 타임스텝마다 바뀐다는 점이다. 행동은 정의할 수 있는 선택의 폭이 크지 않아 비교적 쉽게 정의할 수 있지만, 상태는 정의할 수 있는 방법이 다양하다. 이 상태를 어떻게 정의하느냐에 따라 모델의 성능이 변경될 수 있고, 더불어 확장성을 올바르게 지원할 수 있는지도 결정될 수 있다.

4.1.2 상태

다차원 배낭 문제의 MDP 설계에서 상태에 포함될 수 있는 정보는 다양하다. 예를 들어, 제약 조건의 수 m, 배낭의 초기 용량 C_0^j, 배낭의 남은 용량 C_t^j , 물건의 전체 개수 $|\mathbb{I}_0|$, 남은 물건의 개수 $|\mathbb{I}_t|$, 선택된 물건의 개수 $|\mathbb{I}^{sel}|$, 선택된 물건들 $\mathbb{I}^{sel}$, 각 물건의 가치 v^i, 각 물건의 각 제약 조건 속성의 크기 r^{ij} 등 다양한 정보가 있다. 이때, 상태에 포함되는 정보를 최소화하여 상태 공간의 크기를 줄임과 동시에 효율적인 학습이 가능하도록 그러한 정보를 적절하게 선택하는 것이 중요하다.

본 논문은 주어진 물건의 개수와 무관한 함수 $f(\cdot)$를 만들기 위해, 상태를 1차원 형태의 벡터로 표현하지 않고 2차원 형태의 행렬로 정의한다. 문제 P_t^m에 대한 상태 $s(P_t^m)$ 그림 1과 같다.

타임스텝 t에서 상태의 크기는 $n_t \times (m+1)$이 된다. 행렬 $s(P_t^m)$ 내에 위치한 각 행은 물건 $I^i \in \mathbb{I}_t$의 정규화된 가치 정보 및 속성 정보를 나타낸다. 물건 I^i의 가치에 대한 정규화를 위하여 사용된 $v_t^{\max}$ 는 남은 물건의 가치 중 최대값 $\max_{I^i \in \mathbb{I}_t} v^i$이다. 각 물건의 가치를 $v_t^{\max}$로 나눔으로서 물건들의 가치 스케일 차이에 영향을 받지 않도록 할 수 있다. 또한, 각 물건의 제약 조건 속성의 크기를 배낭 내 해당 속성의 남은 용량 C_t^j로 나눈다. 이를 통해, 각 물건이 얼마나 가치 있는지와 해당 물건을 배낭에 넣을 때 배낭을 얼마나 차지하는지를 더욱 정확하게 파악할 수 있다.

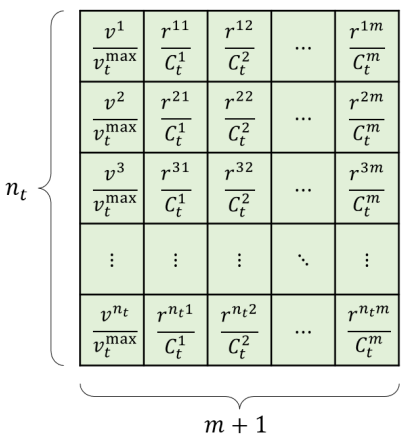


그림 1. 다차원 배낭 문제의 상태
Fig. 1. State of the MKP

4.1.3 보상

MDP 설계에서 보상은 다차원 배낭 문제의 목적과 밀접한 관련이 있다. 본 논문에서는 배낭 문제의 목적이 최대 가치를 얻는 것이므로 매 타임스텝마다 선택한 물건의 가치 v^{sel}를 보상으로 받는다.

이미 선택된 물건이나 제약 조건 때문에 배낭에 넣을 수 없는 물건을 선택하면 보상이 없거나 음의 보상을 받는 것을 고려할 수도 있다. 하지만, 이와 같은 행동들은 액션 마스킹(Action Masking)으로 제외하여 선택할 수 없게 하여 고려하지 않는다.

지금까지 정의한 MDP는 현재 상태 $s(P_t^m)$에서 행동 $a_t = I^{sel}$을 결정하는 데에 과거의 상태 $s(P_{t-1}^m)$와 행동 a_{t-1}이 영향을 미치지 않는다. 이로써, 다차원 배낭 문제를 순차적 의사 결정 문제로 모델링하여 물건의 개수와 무관하게 학습 가능한 MDP를 설계했고, 이러한 MDP 설계는 마르코프 속성을 만족한다. 제안하는 MDP 설계를 활용하면 다양한 규모의 배낭 문제에 대해 유연하게 대응할 수 있다.

4.2 사용하는 알고리즘

앞서 정의한 MDP를 학습하기 위해 강화학습 알고리즘을 선택하고 적용해야 한다. 해당 MDP에서 에이전트의 행동은 n_t개의 물건 중 하나를 선택하는 이산 행동(Discrete Action)이다. 이산 행동을 다루는 강화학습 알고리즘은 기본적으로 DQN 알고리즘이 널리 사용된다. 본 논문에서 강화학습 알고리즘의 학습 절차에 대한 수정은 고려하지 않기 때문에 DQN 알고리즘[5]을 사용한다. DQN 알고리즘에는 여러 변형이 있는데, 본 논문에서는 여러 가지 DQN 변형 중 신경망 구조의 변경 없이 손실 함수만을 변형한 Double DQN 알고리즘[19]을 사용한다. 여러 앞선 연구들이 대체적으로 Double DQN 알고리즘 성능이 DQN 알고리즘 성능을 앞서는 것으로 보고하고 있다.

4.3 어텐션 메커니즘 기반 신경망

앞서 MDP를 정의하면서 언급한 규모 불변성을 활용하기 위하여, 현재 주어진 물건의 개수와 무관하게 학습할 수 있는 신경망이 필요하다. 물건의 개수를 시퀀스의 길이처럼 다루어 시퀀스 모델을 사용하는 것을 고려할 수 있다. 이때, 적용할 수 있는 신경망은 순환 신경망, 포인터 네트워크, 어텐션 메커니즘, 그래프 합성곱 신경망 등이 있다. 이 중 순환 신경망과 포인터 네트워크는 입력 시퀀스의 순서에 따라 결과가 달라지므로 원래 물건 간의 순서가 고려되지 않는 배낭 문제에 적합하지 않다. 그래프 합성곱 신경망은 입력으로 그래프를 받아야 하는데 배낭 문제의 물건 들은 서로 명시적인 관계를 가지지 않으므로, 그래프의 구성 요소인 노드와 간선을 정의하기가 적합하지 않다. 어텐션 메커니즘은 자연어 처리와 같은 시퀀스-투-시퀀스(Seqence-to-Sequence) 문제에서 자주 사용되지만, 어텐션 메커니즘 자체만으론 입력의 순서와 특정하기 어렵기 때문에, 포지셔널 인코딩(Positional Encoding)과 마스킹(Masking) 기법을 통해 입력의 순서를 고려할 수 있다. 달리 말하면, 포지셔널 인코딩과 마스킹이 없는 어텐션 메커니즘은 입력의 순서와 무관하게 사용할 수 있다. 어텐션 메커니즘은 입력 시퀀스의 각 요소를 반영하여 결과를 계산하기 때문에 배낭 문제에서 모든 물건을 고려하여 현재 물건의 선택 여부를 결정하는 데에 적합하다.

본 논문에서 사용하는 어텐션 메커니즘은 시간에 따른 일련의 데이터들을 처리하는 것이 아니라, 하나의 데이터를 일정한 규칙으로 분할하여 시간과 관계없이 처리하는 방식이다. 이 방식은 각 아이템의 정보를 입력으로 받아 그 아이템의 선택 여부를 출력하는 신경망으로 사용되며, 키, 쿼리, 값 모두 동일한 데이터를 사용한다.

그림 2는 강화학습을 활용하여 배낭 문제를 해결하는 기존 논문[12,13]과 본 논문의 상태 정의와 신경망 활용법을 비교한 그림이다. 기존 연구는 하나의 상태 정보를 완전 연결 신경망에 입력으로 넣어 어떤 물건을 선택할지를 결정한다. 본 논문은 하나의 상태 정보를 n_t개의 요소를 가지는 순서가 없는 시퀀스로 분할하여 신경망에 입력으로 주어 어떤 물건을 선택할지를 결정한다. 이를 통해, 고려해야 할 물건의 개수와 독립적으로 상태 정보를 처리할 수 있으며, 물건의 개수가 달라져도 신경망을 그대로 사용할 수 있다.

자연어 처리에서는 어텐션 메커니즘을 시간 순서에 따른 일련의 데이터를 처리하는 데에 사용하며, 입력 데이터의 순서가 중요하므로 포지셔널 인코딩과 마스킹을 사용하여 입력 데이터의 순서를 부여한다. 반면, 본 논문에서는 하나의 데이터를 n_t개의 요소가 있는 시퀀스로 분할하여 어텐션 메커니즘의 입력으로 사용한다. 이때, 입력 데이터의 순서를 고려하지 않아야 하므로 포지셔널 인코딩과 마스킹 기법을 사용하지 않는다.

4.4 모델 아키텍처

제안하는 신경망은 사실 어텐션 메커니즘으로만으로 구성되어있지 않다. 어텐션 메커니즘은 구조상 입력 데이터의 형태와 출력 데이터의 형태가 동일하여 데이터의 형태를 바꿀 수 없다. 따라서, 입력 데이터의 형태를 바꾸는 신경망 층을 어텐션 메커니즘 앞, 뒤에 추가하여 입출력 데이터의 형태를 적절한 임베딩(Embedding) 벡터로 바꾸어야 한다. 또한, 규모 불변성을 활용하기 위해 데이터의 형태를 바꾸는 신경망 층도 물건의 개수에 영향을 받지 않도록 설계해야 한다. 순서가 없는 시퀀스 데이터에서 시퀀스 요소의 특성을 특정 차원으로 임베딩하는 신경망으로 1차원 합성곱 신경망(1D Convolutional Neural Network)을 사용할 수 있다.

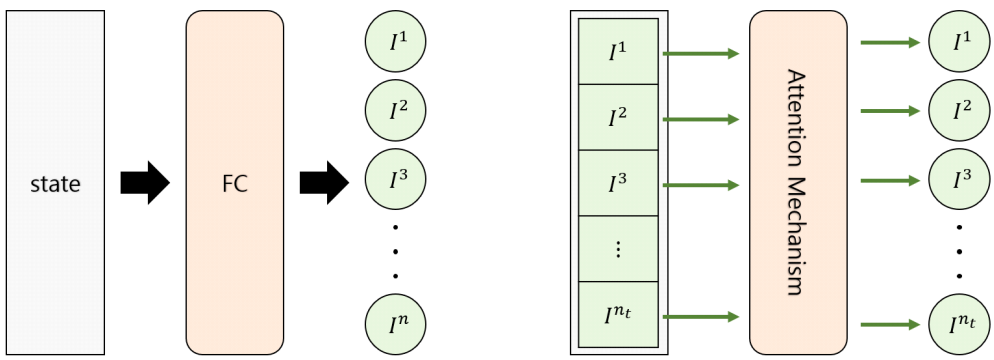


그림 2. 기존 연구와 제안하는 방법의 신경망 구조 비교
Fig. 2. Comparison of Neural Network Structures between Existing Research and Proposed Method

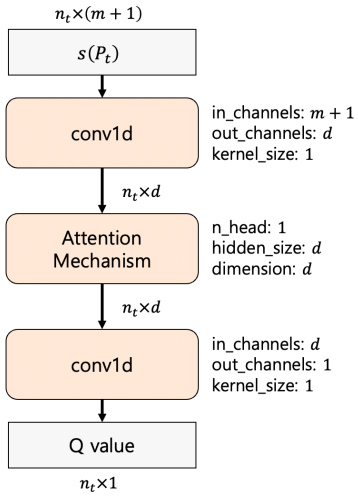


그림 3. 레이어 구성
Fig. 3. Layer Configuration

어텐션 메커니즘과 1차원 합성곱 신경망을 활용하여 Q 신경망을 구성한다.

전체적인 레이어 구성은 그림 3과 같이 구성된다. n_t개의 요소에 대하여 $m+1$차원의 특성을 1차원 합성곱 신경망을 통해 d차원으로 임베딩한다. 임베딩된 d차원 벡터 n_t개로 이루어진 시퀀스를 어텐션 메커니즘의 입력으로 사용한다. 이를 통해 각 물건에 대하여 다른 물건과 비교하여 물건의 중요도를 계산한다. 어텐션 메커니즘의 출력은 n_t개의 요소에 대하여 d차원의 특성을 가지는 시퀀스이다. 이를 다시 1차원 합성곱 신경망을 통해 1차원으로 임베딩한다. 이를 통해, 각 물건에 대한 최종적인 중요도를 계산한 벡터를 얻을 수 있으며, 이 중 중요도가 가장 높은 물건을 선택한다.

V. 실 험

본 논문이 제안하는 MDP와 신경망을 통한 강화학습 기반의 다차원 배낭 문제 해결 방법의 성능을 검증하기 위해 본 장에서 여러 실험을 수행하고 그 결과를 제시한다.

임의의 다차원 배낭 문제 $P^m = (\mathbb{I}, \mathbb{K})$를 구성하는데 필요한 요소들은 다음과 같이 생성하였다. 각 물건의 가치 v^i는 1부터 100까지의 이산균등분포(Discrete Uniform Distribution)에서 임의로 샘플링되며, 각 물건의 각 제약 조건 속성 r^{ij}도 10부터 100까지의 이산균등분포에서 임의로 샘플링되었다. 또한, 배낭의 각 제약 조건 속성에 대한 용량 C^j는 물건의 개수 n에 따라 $25 \times n$으로 설정하였다.

앞에서 제안한 MDP와 Double DQN을 통해 함수 $f(\cdot)$를 학습시켰다. 그리고 함수 $f(\cdot)$가 도출하는 해가 최적해와 얼마나 가까운지를 비교하여 제안한 MDP 가 최적해를 찾을 수 있도록 잘 설계되었는지 확인한다. 그다음 제안한 신경망 구조가 실제로 일반화 성능을 보이는지 확인한다. 제약 조건 속성수는 5, 10개, 물건의 개수는 10, 20, 50, 100개에 대하여, 각 학습에 사용된 문제 인스턴스 수는 최대 $40{,}000 \times n$이다. 학습 과정에서 성능이 일정 기간 동안 나아지지 않으면 학습을 조기 종료하였다. 학습이 완료된 모델의 추론 성능을 확인하기 위해 1,000개의 문제 인스턴스를 생성하여 추론을 수행하고 결과의 평균값을 제시한다.

표 1, 2는 각각 제약 조건 속성수가 5, 10개일 때의 실험 결과를 보여준다. Optimal은 Google OR-Tools 라이브러리[20]로 최적해를 구한 결과이고 Ours는 본 논문에서 제안하는 방법으로 해를 구한 결과이다. OR-Tools 라이브러리는 정수계획법 기반 지역 탐색으로 최적해를 구한다[21]. v^{sum}은 배낭에 담긴 물건들의 가치 합, 즉 $\Sigma_{i \in \mathbb{I}} v^i$을 의미한다. time은 계산 시간을 의미하며 단위는 msec.이다. v_*^{sum}은 최적해의 v^{sum}이다. v^{sum}/v_*^{sum}은 각 방법으로 계산된 해의 v^{sum}이 v_*^{sum}에 얼마나 근접하는지를 나타낸다. 표에서 Ours의

표 1. 실험 결과 ($m=5$)
Table 1. Experiment result ($m=5$)

Method		Optimal	Ours
n=10	v^{sum}	281.3	274.3
	time	11.32	1.03
	v^{sum}/v_*^{sum}	100%	97.5%
n=20	v^{sum}	628.0	615.2
	time	35.73	2.34
	v^{sum}/v_*^{sum}	100%	98.0%
n=50	v^{sum}	1697.3	1673.3
	time	63.30	7.43
	v^{sum}/v_*^{sum}	100%	98.6%
n=100	v^{sum}	3491.7	3440.6
	time	126.65	21.68
	v^{sum}/v_*^{sum}	100%	98.6%

표 2. 실험 결과 ($m=10$)
Table 2. Experiment result ($m=10$)

Method		Optimal	Ours
$n=10$	v^{sum}	255.7	249.2
	time	19.23	0.91
	v^{sum}/v_*^{sum}	100%	97.5%
$n=20$	v^{sum}	596.3	582.9
	time	66.53	2.61
	v^{sum}/v_*^{sum}	100%	97.8%
$n=50$	v^{sum}	1645.3	1615.8
	time	102.41	7.39
	v^{sum}/v_*^{sum}	100%	98.1%
$n=100$	v^{sum}	3394.6	3342.6
	time	220.86	21.42
	v^{sum}/v_*^{sum}	100%	98.4%

v^{sum}/v_*^{sum}는 약 98%로 나오며, 이를 통해 제안하는 MDP가 문제를 해결할 수 있도록 잘 설계되었다는 것을 알 수 있다. 그리고 Optimal과 Ours의 time을 비교해보면 계산 시간이 훨씬 짧다. m이 5에서 10으로 늘어나면서 Optimal의 경우는 계산 시간도 함께 증가하지만, Ours는 그렇지 않다.

심층 신경망을 활용하여 배낭 문제를 해결하는 이전 연구들은 물건의 개수가 바뀌면 신경망을 새로 학습해야 하지만, 본 논문은 제안하는 방법은 새로 학습할 필요가 없다. 제안하는 방법을 사용하면 배낭 문제의 규모 불변성을 활용하여 n이 작은 문제를 쉽고 빠르게 학습한 다음 n이 큰 문제를 해결할 수 있다.

표 3은 물건의 개수와 독립적으로 설계된 MDP와 신경망의 장점을 보여준다. $m=10$ 및 $n=20$인 다차원 배낭 문제를 제안하는 방법으로 학습하고, 동일한 m에 대하여 다양한 n을 가진 다차원 배낭 문제를 풀었을 때의 성능 및 추론 시간을 나타낸다. 시간의 단위는 msec.이다. 결과를 보면 분명 $n=20$인 문제만 학습했음에도 불구하고, n이 20인 문제뿐만 아니라 n이 50, 100, 200, 300인 문제를 해결하는 것을 알 수 있다. 게다가, 성능이 전혀 떨어지지 않고 꾸준히 유지되는 것을 볼 수 있다. 제안하는 방법을 통해 1시간 미만의 학습 시간만으로 n이 300인 문제를 정확한 알고리즘보다 훨씬 짧은 시간 내에 최적에 가까운 해를 구할 수 있다. 그리고 일반화 성능을 보았을 때, 규모가 더 큰 문제도 풀 수 있는 잠재력을 가진 것을 알 수 있다.

Ⅵ. 결 론

배낭 문제에서 물건의 개수가 바뀌는 일은 쉬운 일이지만, 이전까지의 강화학습 기반 배낭 문제 해결에 대한 연구는 물건의 개수에 종속적인 방법을 제안하였다. 이러한 방법들은 물건의 개수가 바뀌게 되면 모델을 새로 학습해야 하므로 물건의 개수가 바뀌는 일이 자주 발생하는 배낭 문제에서는 사용하기 어렵다.

본 논문에서는 물건의 개수가 바뀌는 배낭 문제에도 적용 가능한 강화학습 기반 다차원 배낭 문제 해결 방법을 제안한다. 이 방법은 물건의 개수에 종속적이지 않아 물건의 개수가 바뀌어도 모델을 다시 학습할 필요가 없다. 배낭 문제의 규모 불변성을 활용하기 위해, 물건의 개수와 독립적인 상태를 정의하고 학습 가능한 MDP를 제안했다. 게다가 이 MDP를 물건의 개수와 무관하게 사용 가능하도록 어텐션 기반 네트워크를 제안했다.

실험을 통해 MDP와 신경망의 성능을 분석한 결과, 배낭 문제를 효과적으로 해결할 뿐만 아니라 일반화 성능도 뛰어난 모습을 보인다. 향후에는 본 논문에서 제안한 방법에 대하여 재사용성, 확장성, 일반성 관점에서 그 성능을 더욱 높일 수 있는 방안을 연구할 예정이다.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J.

표 3. $n=20$인 문제를 학습한 ours의 일반화 성능($m=10$)
Table 3. Generalization performance of ours trained on $n=20$ problems ($m=10$)

	n=20	n=50	n=100	n=200	n=300
v^{sum}/v_*^{sum}	97.6%	98.3%	98.7%	98.8%	98.8%
time	2.68	8,.02	23.01	68.74	146.51
Optimal's time	63.12	99.86	209.44	857.52	4254.23

Schrittwieser, et al., “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, pp. 484-489, Jan. 2016. (https://doi.org/10.1038/nature16961)

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, et al., “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, pp. 354-359, Oct. 2017. (https://doi.org/10.1038/nature24270)

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, et al., “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” *Science*, vol. 362, pp. 1140-1144, Dec. 2018. (https://doi.org/10.1126/science.aar6404)

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013. (https://doi.org/10.48550/arXiv.1312.5602)

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529-533, Feb. 2015. (https://doi.org/10.1038/nature14236)

[6] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell, “Agent57: Outperforming the atari human benchmark,” in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, pp. 507-517, Jul. 2020. (https://doi.org/10.48550/arXiv.2003.13350)

[7] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, “Neural combinatorial optimization with reinforcement learning,” *arXiv preprint arXiv:1611.09940*, 2016. (https://doi.org/10.48550/arXiv.1611.09940)

[8] H. Dai, E. Khalil, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, pp. 6351-6361, Dec. 2017. (https://doi.org/10.48550/arXiv.1704.01665)

[9] W. Kool, H. van Hoof, and M. Welling, “Attention, learn to solve routing problems!,” *arXiv preprint arXiv:1803.08475*, 2018. (https://doi.org/10.48550/arXiv.1803.08475)

[10] M. Jalali Varnamkhasti, “Overview of the algorithms for solving the multidimensional knapsack problems,” *Advanced Studies in Biology*, vol. 4, no. 1, pp. 37-47, 2012.

[11] S. Gu and T. Hao, “A pointer network based deep learning algorithm for 0–1 Knapsack Problem,” *ICACI*, pp. 473-477, Jun. 2018. (https://doi.org/10.1109/ICACI.2018.8377505)

[12] R. R. Afshar, Y. Zhang, M. Firat, and U. Kaymak, “A state aggregation approach for solving knapsack problem with deep reinforcement learning,” in *Proc. 12th Asian Conf. Mach. Learn.*, vol. 129, pp. 81-96, Nov. 2020. (https://doi.org/10.48550/arXiv.2004.12117)

[13] B. Yildiz, “Reinforcement learning using fully connected, attention, and transformer models in knapsack problem solving,” *Concurrency and Computation: Practice and Experience*, vol. 34, no. 9, pp. e6509, Apr. 2022. (https://doi.org/10.1002/cpe.6509)

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst*, pp. 6000-6010, Dec. 2017. (https://doi.org/10.48550/arXiv.1706.03762)

[15] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, pp. 2692-2700, Dec. 2015. (https://doi.org/10.48550/arXiv.1506.03134)

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, et al., “Asynchronous methods for deep reinforcement learning,” in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48, pp. 1928-1937, Jun. 2016.

(https://doi.org/10.48550/arXiv.1602.01783)

[17] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *J. Heuristics*, vol. 4, pp. 63-86, Jun. 1998. (https://doi.org/10.1023/A:1009642405419)

[18] S. Hanafi and A. Freville, "An efficient tabu search approach for the 0 - 1 multidimensional knapsack problem," *Eur. J. Operational Res.*, vol. 106, no. 2-3, pp. 659-675, Apr. 1998. (https://doi.org/10.1016/S0377-2217(97)00296-8)

[19] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 30th AAAI Conf.*, pp. 2094-2100, Feb. 2016. (https://doi.org/10.48550/arXiv.1509.06461)

[20] L. Perron and V. Furnon, OR-Tools(Google), Retrieved Mar. 13, 2023, from https://developers.google.com/optimization/

[21] J. Hwang, "Integer programming-based local search techniques for the multidimensional knapsack problem," *J. Korea Soc. Comput. and Inf.*, vol. 17, no. 6, pp. 13-27, Jun. 2012. (https://doi.org/10.9708/JKSCI.2012.17.6.013)

최 요 한 (Yohan Choi)

2021년 8월 : 한국기술교육대학교 학사

2023년 8월 : 한국기술교육대학교 석사

<관심분야> 딥러닝, 강화학습, 조합최적화, 제로샷 학습

[ORCID:0009-0003-6861-3030]

석 영 준 (Yeong-Jun Seok)

2022년 2월 : 한국기술교육대학교 학사

2022년 3월~현재 : 한국기술교육대학교 석사과정

<관심분야> 심층강화학습, 조합최적화, 네트워크 슬라이싱, 양자컴퓨터

[ORCID:0009-0007-6942-3596]

김 주 봉 (Ju-Bong Kim)

2017년 2월 : 한국기술교육대학교 컴퓨터공학과 학사

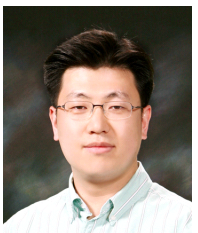
2019년 2월 : 한국기술교육대학교 컴퓨터공학과 석사

2023년 8월 : 한국기술교육대학교 컴퓨터공학과 박사

<관심분야> 정밀 제어, 딥러닝, 멀티 에이전트 강화학습

[ORCID:0000-0001-6406-3092]

한 연 희 (Youn-Hee Han)



1996년 2월 : 고려대학교 수학과 학사

1998년 2월 : 고려대학교 컴퓨터공학과 석사

2002년 2월 : 고려대학교 컴퓨터공학과 박사

2002년 3월~2006년 2월 : 삼성종합기술원 전문연구원

2013년 9월~2014년 8월 : SUNY at Albany, Department of Computer Science 방문교수

2006년~현재 : 한국기술교육대학교 컴퓨터공학부 교수

<관심분야> 사물인터넷, 5G/6G, 딥러닝, 강화학습, 조합최적화

[ORCID:0000-0002-5835-7972]