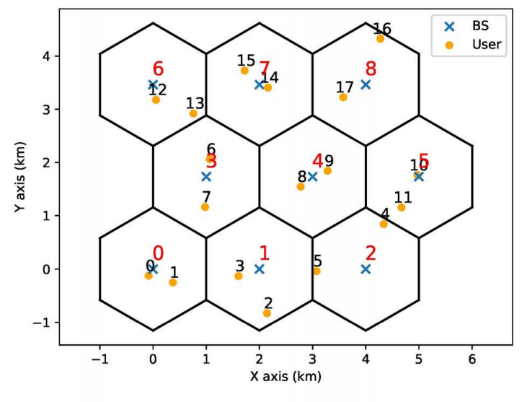
1. power allocation in multi-user cellular networks with deep q learning approach

cited by 23 times, [ICC 2019 - 2019 IEEE International Conference on Communications (ICC)](https://ieeexplore.ieee.org/xpl/conhome/8753818/proceeding)

1. Environment

해결하고자하는 환경 : Distributed dynamic downlink power allocation with multiple users and an interfering multiple-access channel(IMAC)

User가 늘어남에 따라 intra-,inter- cell interference 관리는 중요해졌지만 NP-hard 문제다.

- “Contribution”

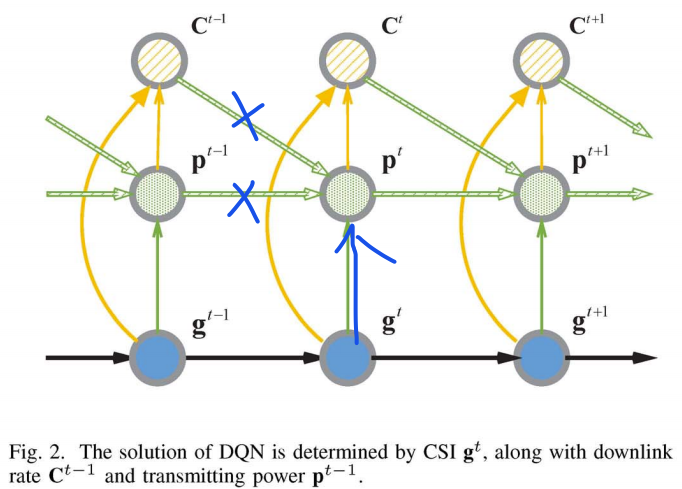
- 1. Model-free two-step learning as known Transfer learning(off-line DRL train in simulated scenarios. -> on-line train)

- 2. No future rewards

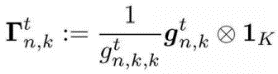
- 3. After centralized training, tested by distributed execution : outperforms model-driven, good generalization

- N cells, BS at the center of each cell, K users sharing frequency band.

- Optimization target : channel gain + small scale fading -> SINR구해서 -> sum data rate(간단한 최적화 모델링)

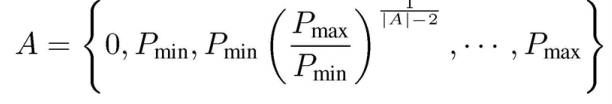
원래라면 CSI에서 optimal solution p star를 바로 구할 수 있지만 퍼포먼스가 낮기 때문에 보충 요소로 C, p를 추가 투입.  
BS-agent link를 agent로 보고 multi-agent로 접근하려했지만 문제가 복잡해져 only one agent만 학습한다. Using all agent’s experience replay memory.

1. State : optimal p를 current state의 CSI g\_t로만 찾기는 힘들어서 c, p라는 보충 요소 등장

* One agent learning하는데,,, SINR구하는데 있어 CSI information을 완벽히 안다고 가정.
* 
* logarithmic normalized interferer set 정의(채널 진폭이 종종 크기 순서에 따라 달라져서 로그 표현이 선호된다고 함.)
* 차원을 줄이기위해 첫번째 I\_c elements(열?)만 놔둠. 즉, input space of DQN = 3I\_c(C, P추가 되었으니)



1. Action : DQN의 action은 discrete해야하기 때문에 A-1 level로 나누어줌



1. Reward : 정교하게 디자인해도 대부분이 suboptimal로 수렴하기 때문에 그냥 downlink를 바로 reward.



1. Deep neural network

* 4 layer feed-forward
* The number of neurons of 2 hidden layers = 128, 64
* Activation function of output = linear, of 2 hidden = ReLU

n : BS, k : user

Independent channel gain(CSI information)

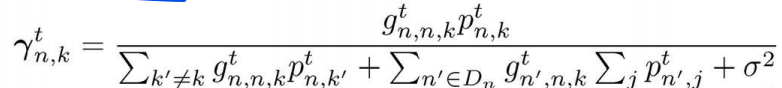
Small scale complex fading element

Large scale fading component

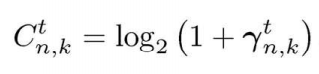
first-order complex Gauss-Markov process



, where J0 = first kind zero-order Bessel function, f\_d = maximum doppler frequency, T\_s = time interval



D\_n = set of interference cells around the n-th cell. P = emitting power of BS, sigma = noise power

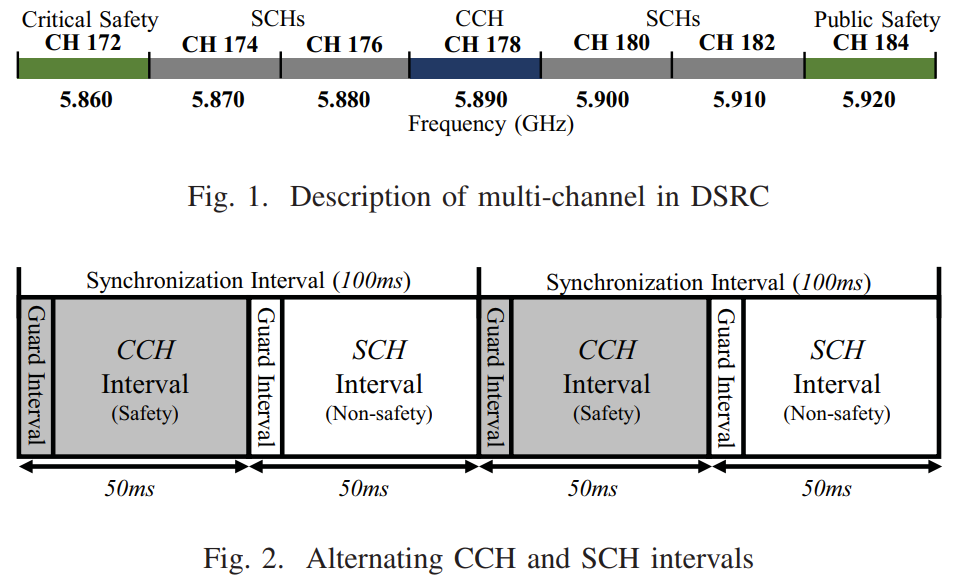
downlink rate of this link.

1. Multiple Channel Access using Deep Reinforcement Learning for congested vehicular Networks

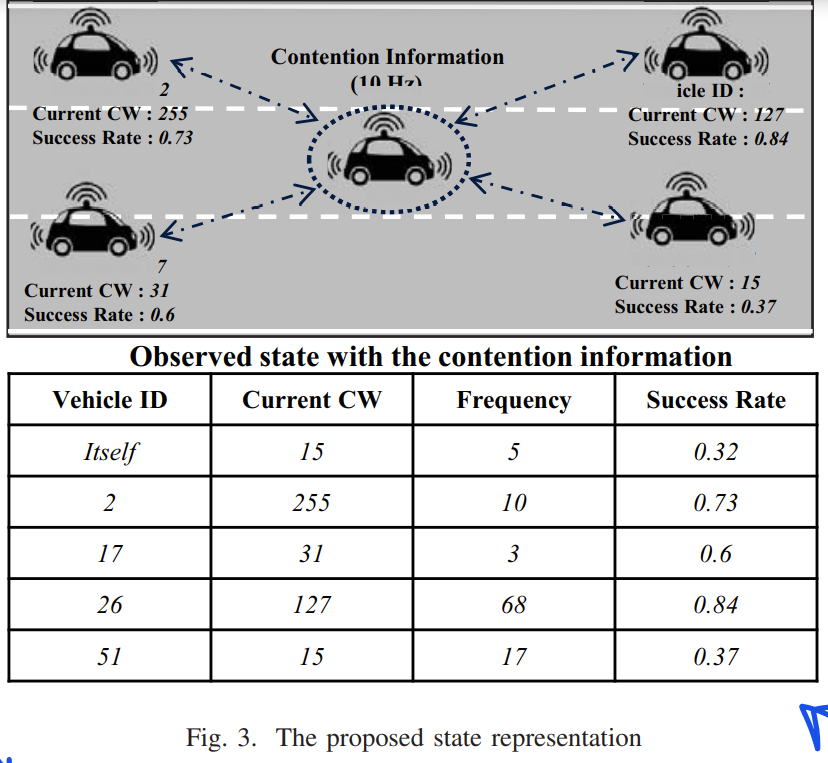
Cited by 2 times, [2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)](https://ieeexplore.ieee.org/xpl/conhome/9121635/proceeding)

1. Environment :

* self-experience-based CW adaptation algorithm employing DRL
* vehicle(agents) broadcast the safety packet using V2V communication and receive transmission results from a VANET. Consequently, vehicles learn to adjust the optimum CW
* 알고리즘의 목적 : high PDR(Packet Delivery Ratio) + low end-to-end delay
* Key features of the algorithm : follows multi-channel operation of DSRC(Dedicated Short Range Communications) standard. + adapt CW following CSMA/CA + contention information-base state 활용
* DSRC의 multi-channel operation : CCH(control-channel), SCH(service-channel)으로 나누어지는데 각 interval = 50ms. 그래서 모든 vehicles은 100ms마다 safety packet 전송(in CCH interval ; CCHI), SCH interval ; SCHI에는 가장 가까운(지정된) vehicle(node)에 unicast ACK 전송. SCHI 때 target vehicle 선택 & transmit ACK



* “Contribution” = contention information-based state representation + the DSRC multi-channel protocol
* State representation : a fully informative state representation(with neighboring vehicles)

CCHI일때, broadcast safety packet. -> 이에 따라, vehicles는 contention information-based state with collected CW values 생성 -> congestion level 간접 추측 + state는 DQN에 의한 function approximation

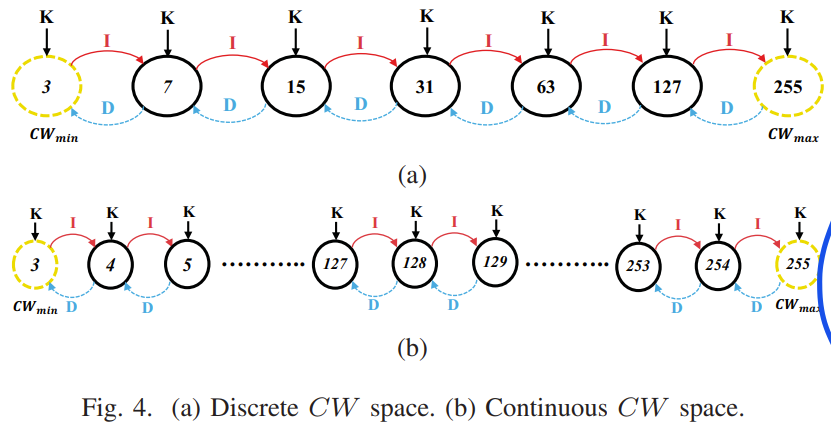
1. state : <CW, F, S>

CW : contention window

F : frequency value

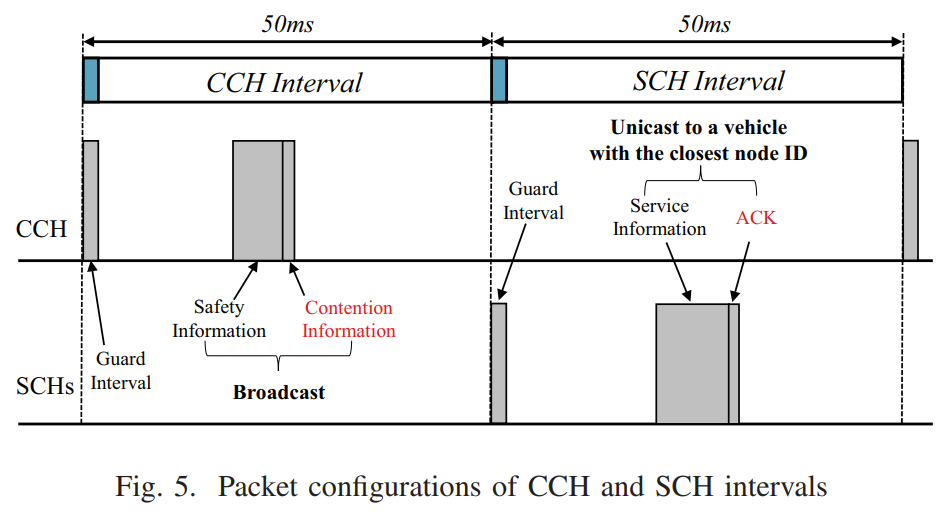
S : success rate

1. action : 3가지(keep, increase, decrease), CW Change를 continuous, discrete 2가지로 표현했는데 뒤의 실험에서 2가지 방법 비교



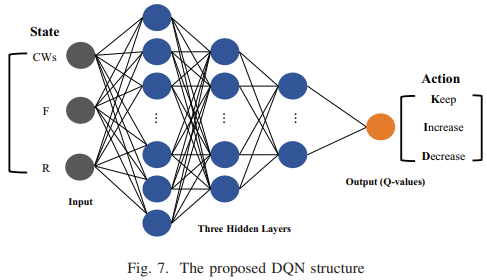
1. reward : broadcast 성공시 +1, 실패시 -1 -> 그래서 SCHI 때 각 agent의 broadcast성공인지 판단 가능

* CW adaptation하기 위해서 all vehicles must receive ACK -> Receiving scheme을 어떻게 구성할 것인 것 -> unicast-based ACK scheme during SCHI -> 이를 통해, its broadcast during SCHI가 성공했는지 판단가능



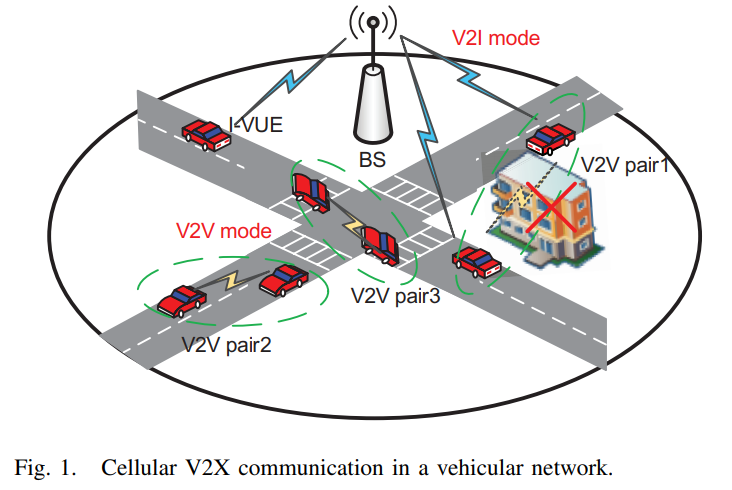
1. Deep neural network

* 3 hidden layers with the number of neurons 256, 128, 64 using Leaky-Relu as activation function.



1. Deep Reinforcement Learning Based Mode Selection and Resource Allocation for Cellular V2X Communications

Cited by 12 times, [IEEE Internet of Things Journal](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6488907) ( Volume: 7, [Issue: 7](https://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=9138535), July 2020)

1. Environment : Resource allocation for V2X(in heterogeneous QoS requirements, unreliable V2V links)

BS : 중앙 위치, single antenna, VUE : random distributed, single antenna

V2I : bandwidth-demanding(대역폭 요구사항이 많은?) 정보 업로드

V2V : safety-critical(안전에 중요한 메시지) 1:1 발송 및 수신, single RB 선택함

V2V link 수가 V2I link 수보다 훨씬 많음

VUE의 높은 이동성 -> large-scale channel gain with path loss and shadow fading considered.

Channel gains = I-VUE – BS, V2V tx – BS, V2V pair

Interfering channel = V2V tx – BS, V2V tx – V2V rx, I-VUE – V2V rx

장애물 -> LOS, NLOS state  
“contribution”

* 1. V2V link의 불확실성을 줄이기 위해, V2I based forwarding mode 사용 -> V2V link pair는 V2V 또는 V2I 모드 선택 가능
* 2. MDP로 모델링, DRL-based decentralized algorithm. Agent = each V2V pair
* 3. Training data가 부족하기 때문에 Two-timescale federated DRL-based algorithm developed = Large timescale(graph-based vehicles 클러스터링) + small timescale(같은 클러스터링 내의 vehicles끼리 강한 DRL model 훈련), global DRL model은 어떤 agent(even newly activated V2V pairs)도 다운로드 후 사용 가능
* 4. Vehicular density와 outage threshold의 영향이 설명됨.

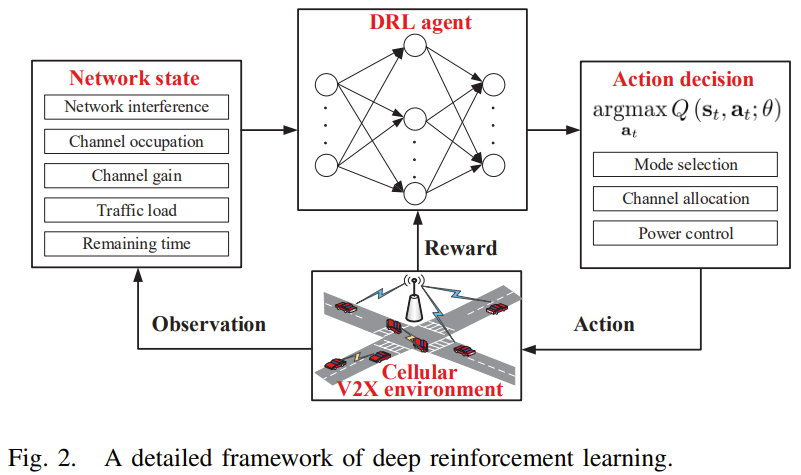
“communication modes for I-VUE”

* Only uplink V2I communication

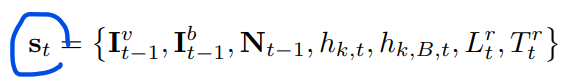
“communication modes for V2V pairs”

* 채널 quality에 따라 mode 선택 가능.-> V2V mode(Interference comes from I-VUE and V2V pairs sharing the same RB.), V2I mode(safety-critical messages가 가장 먼저 BS거쳐 receiver에게 forwarded, BS의 transmit power가 크기 때문에 uplink SINR이 더 작음, unused RBs only can be allocated.)

“QoS requirements of I-VUEs and V2V pairs”

* Capacity requirements of the I-VUEs + Latency and reliability requirements of the V2V pairs.

1. State : 7 parts



I\_v : the received interference power at the V2V receiver

I\_b : the received interference power at the BS

N : the number of selected neighbors on each RBs

h\_k,t : the large scale channel gain from the V2V transmitter to its corresponding V2V receiver

h\_k,B,t : the large scale channel gain from the V2V transmitter to the BS

L : current load

T : remaining time to meet the latency threshold

1. Action

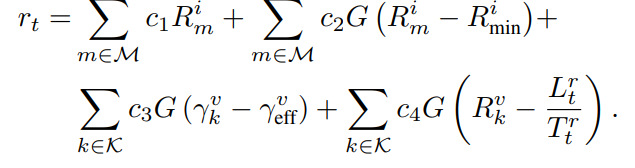


A : the RB allocation

S : communication mode selection

P : transmit power level of the V2V transmitter

1. Reward



1st : sum capacity revenue of I-VUEs

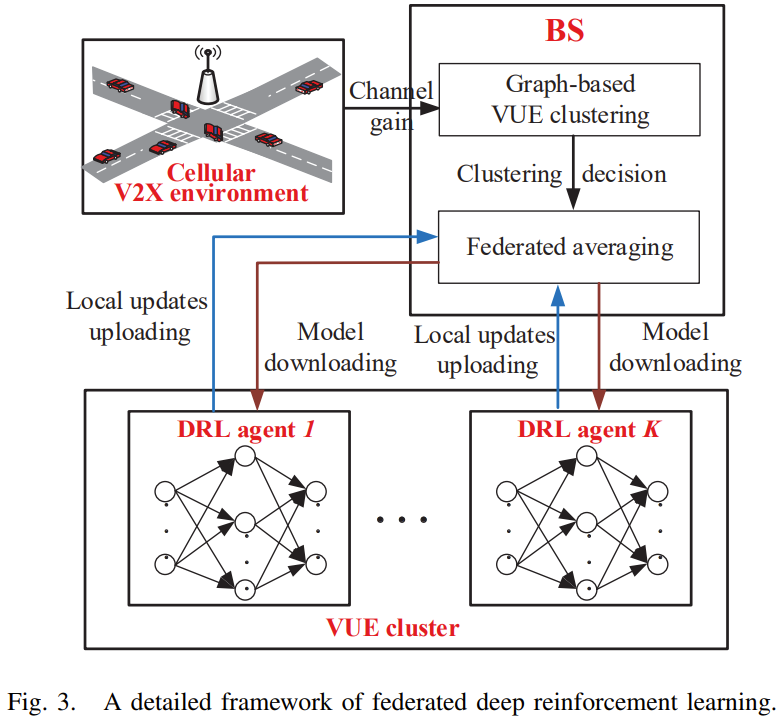
2nd : penalty of unsatisfied capacity for I-VUEs

3rd, 4th : impacts of the reliability and latency requirement

“Federated DRL-based Semi-Decentralized Algorithm”

* Challenges : stringent latency requirement, lack of training data, newly activated V2V pairs, high mobility of vehicles, similar channel quality, environment observations -> drl-based decentralized algotithm쓰면 안되는 이유
* 1. Two time scale federated DRL framework

Federated learning : centralized learning(privacy + communication cost issue) + local learning(time-consuming + imprecise)의 절충안. 즉, 모델 트레이닝과 트레이닝 데이터에 직접 접근할 필요를 분리(Local : local raw data를 활용한 local training + Server : infrequent averaging of local models) -> DRL 성능 향상

Large time scale : BS는 channel gain, groups nearby VUEs with the similar channel gain에 따라 undirected graphs 구성. RB가 각각 cluster에 배정

Small time scale : Federated learning 적용!, V2V pairs 비동기적으로 select actions and train local model -> 수백 subframes마다 local models uploaded and averaged to BS resulting global feedback to whole V2V pairs.

* 2. Centralized VUE Clustering on a Large Timescale

Undirected graph로 모델링되는데, 꼭짓점과 corresponding edge로 모델링함. Large scale channel gain 만 채택함(link between VUES is unreliable), edge에 weight 부여 후 weight 합계 maximize하는 방향으로 clustering 진행(=clustering VUEs with similar channel gain) -> NP-hard(K-means 등으로 안풀림) -> spectral clustering으로 해결(refer[34]참고)

* 3. Federated DRL on a small Timescale

BS distributes pre-trained or averaged model to the V2V pairs in the same clusters -> 각 V2V pair는 DRL-based decentralized algorithm 실시(select their own action without any knowledge of other pairs.) in order to train their own model based on local training data -> BS selects V2V pairs from same clusters to upload their model -> federated averaging(by mini-batch) & redistributes averaged model back until next round.

Local observation이기 때문에 asynchronous scheme 제안됨. 즉, 각 V2V pair는 특정 subframe에 할당되고 비동기적으로 action selection.

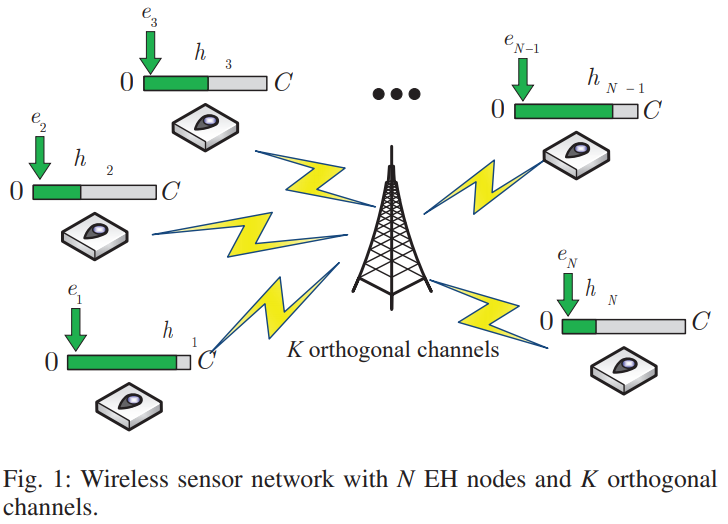
1. Deep neural network

* 1 hidden layer(256), 모두 fully connected layer,

1. Partially Observable Double DQN Based IoT Scheduling for Energy Harvesting

Cited by 3 times [2019 IEEE International Conference on Communications Workshops (ICC Workshops)](https://ieeexplore.ieee.org/xpl/conhome/8751668/proceeding)

1. Environment : EH을 어떻게 조절할 것인가! In POMDP

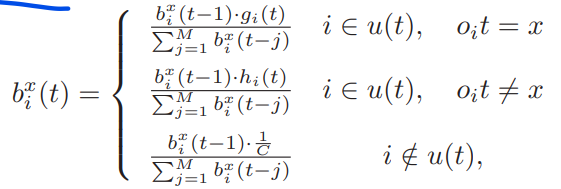
POMDP라 가운데 BS가 node의 상태를 완전히 알 수 없다.

* each node = EH device + rechargeable battery with limited capacity.
* BS only observes power information about “partial” nodes.
* “Contributions”
* 1. Double DQN = To reduce overestimated caused by Q-learning or DQN.
* 2. DDQN + EH wireless in POMDP에서 최초
* Nodes equipped with EH devices and rechargeable battery, BS, K orthogonal channels,
* In a time step, 1 orthogonal channel can be occupied by only one node. Channel gain is constant during time step(TS). Energy arrive is poisson process.
* At the beginning of each TS, BS produces scheduling policy -> broadcasts the policy to all nodes. -> received the information about current power of scheduled node. -> scheduled node는 data 전송 시도 후 다시 residual power를 BS로 전송
* 2 Conditions when transmitting data : its power is more than threshold + it is scheduled in TS.

1. State : POMDP라 BS는 scheduled node에 대해서만 대해서만 학습할 수 있다.

MDP라면 전체 정보를 알기에 이렇게 정의를 하겠지만,,

이렇게 state를 변형한다.

하지만, 이것만 가지고는 optimal policy 구하기 힘드니,, observation of scheduled node를 all nodes로 확장 -> belief state  x : battery capacity, I : node

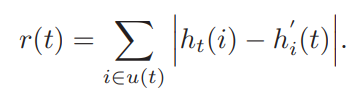
첫 두 줄 ->scheduled node가 어느 상태에 있는지 판단 가능(g\_i, h\_i : To represent the changing trend of the scheduled nodes’ belief state)

Non-scheduled node : 그냥 1/C 곱하기 때문에 어느 상태인지 판단 힘듬

1. Action : scheduling policy(broadcast to all nodes and receive the information about current power)

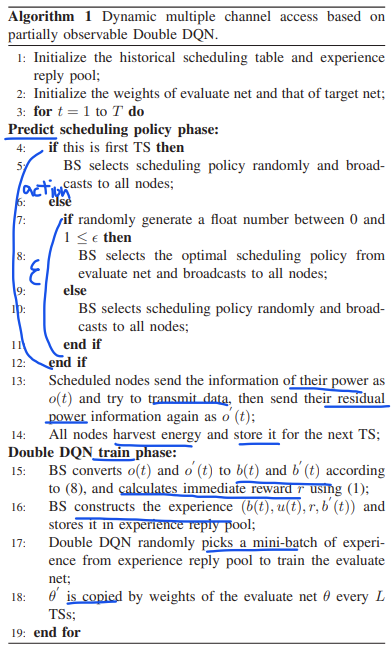
* Dimension of action : K(채널 수) x N(노드 수)

1. Reward

h : the power information about the scheduled node

h` : the residual power to BS again after attempting to transit data

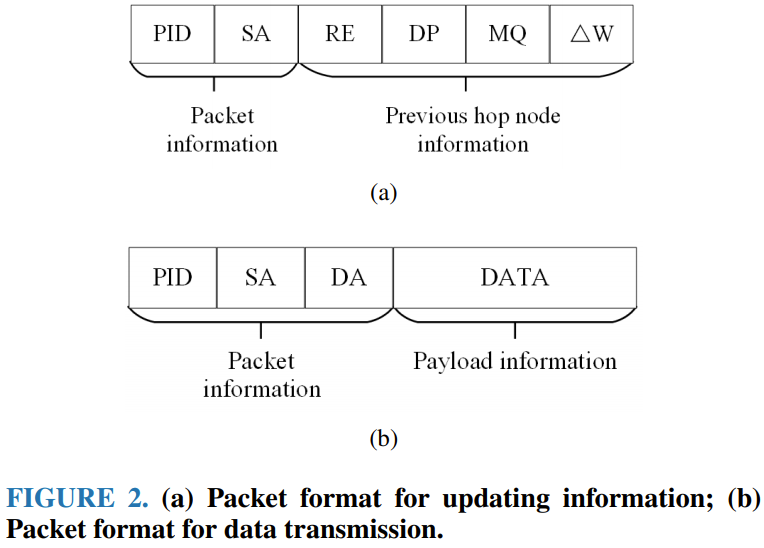
1. Deep neural network(keras in tensorflow)
2. fully connected layer(Sequential), hidden layer의 뉴론 수는 (input 뉴론 + out 뉴론) / 2, ReLU



1. DQELR : An adaptive Deep Q-Network-based energy- and latency-aware routing protocol design for underwater acoustic sensor networks.

Cited by 33 times, [IEEE Access](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6287639) ( Volume: 7), 09 January 2019

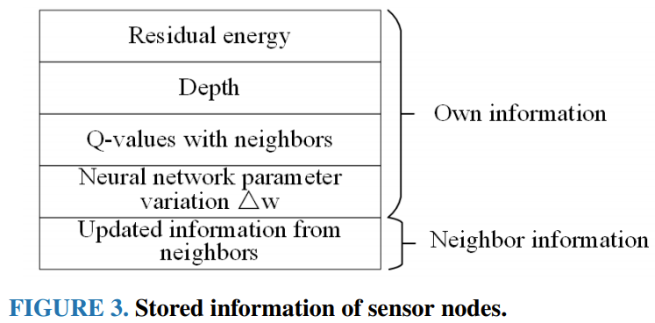
1. Environment

* DQELR(a adaptive deep q-network based energy- and latency-aware routing protocol)을 제안함
* “contribution”
* 1. DQN as a main tech in routing protocol을 통해 UASNs(Underwater Acoustic Sensor Network)에서 network lifetime을 연장함, 이 때 node의 많은 state(residual energy, depth info 등)을 다루는데 에 DQN이 Q-learning보다 적합함
* 2. In UASNs, information interactions through broadcast consumes a great deal of energy -> a hybrid of the broadcast and unicast communication proposed.
* 3. Routing decisions according to residual energy under the premise of strictly limiting end-to-ed latency -> can prolong network lifetime(routing decision could be changed -> how?)
* 4. On policy training -> update Q-values and correct network parameter
* 5. Extensible : reward function 계산할 때, other factors such as node density and environmental noise 등을 합칠 수 있다.
* “Protocol overview”
* Source nodes -> relay nodes -> sink nodes : collected data packets transmitted
* Agent : each packet
* After sending a packet, each agent can receive a reward that updates a new state.
* Each node can obtain the information necessary to calculate Q-values through broadcasting -> the node can make a optimal decision when it needs to send a packet -> energy consumption reduced
* “DQELR Protocol mechanism”
* 1. A hybrid -> broadcast : to update information, unicast : to transmit data
* 2. When making routing decision, both off-policy and on-policy are used.
* 3. In the case of dynamic topology in UASNs, on-policy adopted to make a new routing decision
* 4. Assumption = sensor node can get their own residual energy, depth information + topology changeable but relatively stable in short time + computational delay neglected.
* State(residual energy, depth etc.) assumed -> DQN off-policy to make routing decisions -> sensor network deployed -> when running underwater, network topology and state changed -> on-policy adopted -> broadcast & unicast to decide next optimal forwarder
* In addition, when chosen optimal node is out of range, suboptimal Q-value can be chosen as forwarder. On policy method to make a new routing decision in changing environment.
* “Packet format”

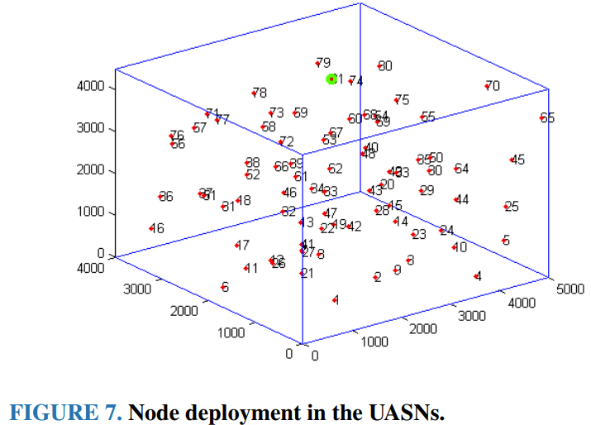
1. Updating information

* PID : Packet ID, SA : source address, RE : residual energy, DP : depth, MQ : maximum Q-value with optimal next forwarder, Gradient w : parameter about the loss function.
* PID, SA : permanent, others : can be changed in the course of transmission
* The stage when the optimal next forwarder has been calculated.

1. Data transmission : Just carry the packet.

* DA : destination address
* For routing decision, store the information defined in Fig. 3 -> calculate Q-values.

* In order to reduce overheads, the information updating stage which causes the overhead is designed to occur periodically. -> overhead neglected, computational overhead neglected.
* “DQN-based routing decision algorithm” : neural network model
* Feature extraction 선행되어야함 : preprocessed as the input for the neural network -> residual energy, depth
* “DQN-based routing decision algorithm” : Off-policy training(Before deploying sensor nodes)
* 1. Initialize network parameter, initialize reward, Q-value according to given tuple
* 2. Loss and new Q-values needs to be stored in an experience pool to be updated at an appropriate time.
* “DQN-based routing decision algorithm” : On-policy training
* Asynchronous method in the one-step DQN : when the loss value of a tuple is obtained, not update immediately, but update after certain period concentratively -> meets the requirement of noncorrelation among the inputs of neural network.

80 sensor nodes, 1 sink

1. State

* Residual energy and depth of A is 1000J, 500m respectively, when max energy is 1000J, max depth is 800m(neighbor node인 B의 값도 다 알고 있다고 가정). Then, if the information is transmitted from A to B, the tuple(A, 1, 0.625, B) is the set of state and action.

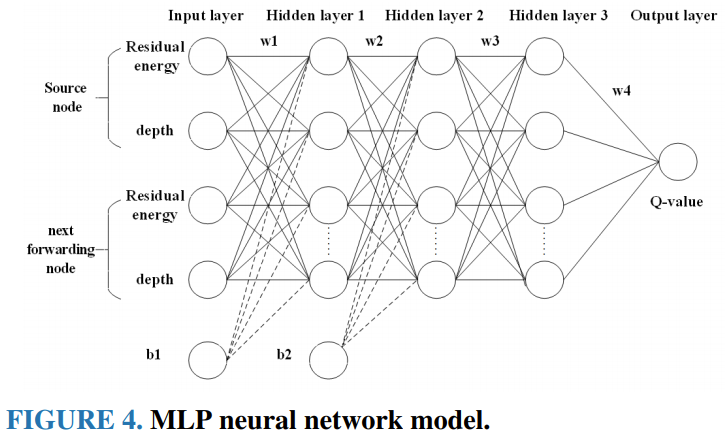
1. Action

* Forwarding a packet

1. Reward

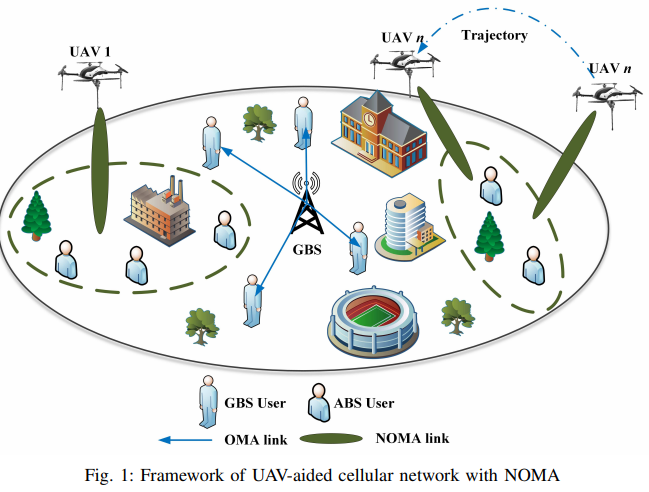
* If node transmits the packets to sink, reward equals 100; otherwise, -> 100이라는 reward가 훨씬 커서 node가 sink에 가까운 route 고르도록 유도

1. Deep neural network



1. Multi-Agent Reinforcement Learning in NOMA-aided UAV Networks for Cellular Offloading

Cited by 0 times, [arXiv.org](https://arxiv.org/)

1. Environment

야외 down-link, 유저 많음, 가운데 GBS

UAV : single 안테나, NOMA(intra-cell 간섭 영향받음)

UAV : GBS와는 다른 주파수대역씀

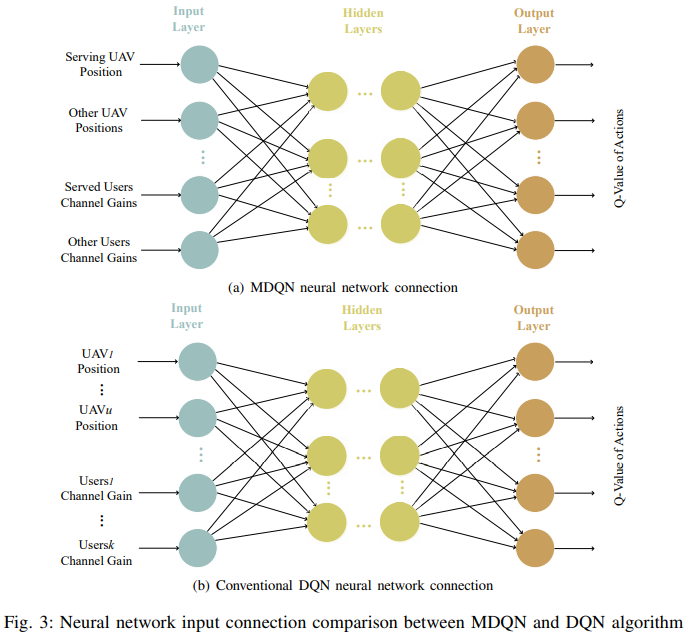
유저 : random roaming + directional walking

.Cluster : All user served, 반복 x, helpful to reduce 간섭

UAV : 유저 위치 체크, re-clustering

최적화 : user clustering + optimization for trajectory and power allocation

online  
여기서 특이한 점이, nn을 모든 agent가 공통모델로 학습하는데 1번 agent가 nn에 연결되면 나머지 agent에게는 제한을 건다.(아래 그림 참조)



1. State



L\_u : connecting agent(UAV)의 3차원 좌표

L\_s : 다른 UAV들 좌표(inter-cluster 간섭 원인)

g\_u : 연결된 유저들의 channel gain

g\_s : 다른 UAV와 연결된 유저들의 channel gain

1. Action

* Movement action space : 7개(수평 왼, 수평 오, 수평 앞, 수평 뒤, 수직 위, 수직 아래, 그대로)

If out of bound, action default is hover

* Power allocation action space : multiple gears

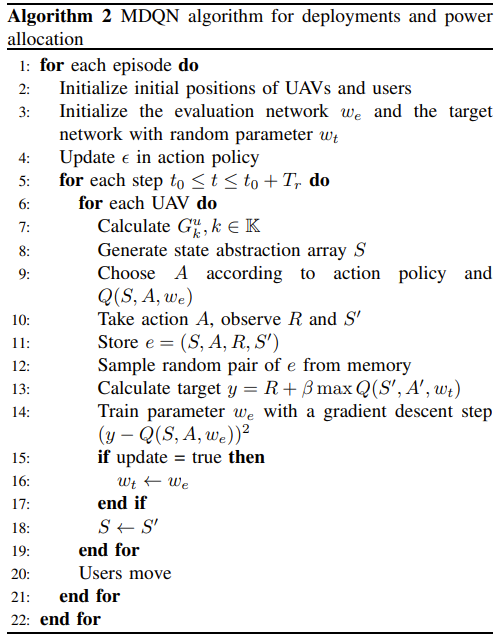
1. Reward : total throughput under constraints



R : sum data rate(multi agent니까 리워드를 total sum으로 설정!)

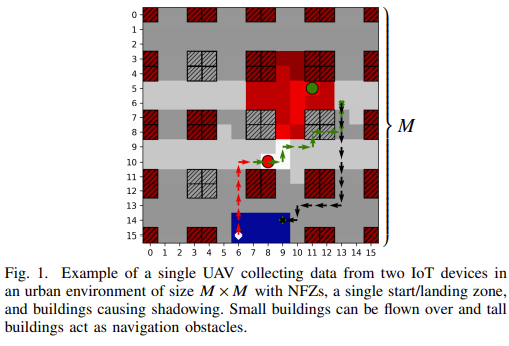
Gamma : penalty coefficient(QoS 최대한 보장키 위한 수단)

1. Deep neural network

* 3 layers (a 40 nodes hidden layer), ReLU, MSE, Adam Optimizer
* 

1. Multi-UAV Path Planning for Wireless Data Harvesting with Deep Reinforcement Learning

Cited by 1 times, [arXiv.org](https://arxiv.org/)

1. Environment

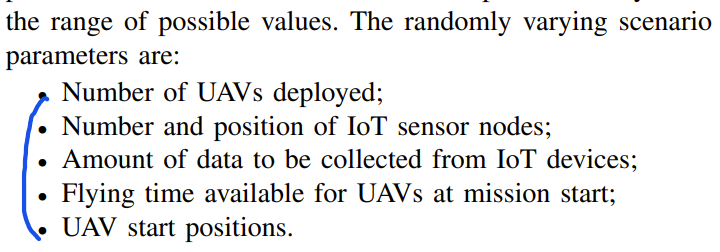
Square grid world = start/landing position + position UAV cannot occupy + Obstacles blocking wireless link

A team of UAV(동일한 UAV)의 path planning problem : Dec-POMDP

제약 : trajectory + battery + urban environment + wireless(random signal blocking events)

“Contribution” : 시나리오 파라미터에 여유를 주는 일반화가 가능한 DRL method by using “centered global-local map processing”

* Flying time의 constraint : Dec-POMDP with full reward function description
* Dec-POMDP를 Deep multi-agent RL로 해결
* Dual global-local map processing : 큰 맵과 state spaces에 대한 학습과 적응 효율성에 있어 이점 using map centering!
* Parameter generalization : the learned policy can be reused over a wide array of scenario parameters.



* “UAV model”
* UAV의 state = 3D position + operation status + battery level
* UAV의 action = 6가지, 1 time slot에 움직이는 거리 = cell size
* “Link performance model”
* Communication time slot이 mission time slot보다 작게 정의해서 활용
* IoT Device sensor has a finite amount of Data
* UAV-to-ground channel model : links with LOS/NLOS + path loss + shadow fading
* “Multiple access protocol”
* TDMA : a UAV to various ground user, inter-UAV interference는 없음, IoT device는 multi-band node로 작동해서 all UAV와 통신 가능(Scheduling decision은 action space에 속하지 않음
* Dec-POMDP = state space + joint action space + transition probability function + reward function + joint observation space + observation function + discount facto
* “Map-processing”
* 1단계 = agent 위치 중심으로 map을 centering(장 : CNN 사용 가능?, agent의 action은 상대적 위치 e.g. its distance to sensor devices, 단 : map size, observation space 늘어남,)
* 2단계 = centered map을 compressed global + uncompressed but cropped local 2가지로 표현 -> 필요한 뉴럴넷 사이즈 줄일수 있음.(distant feature의 디테일 수준은 close object보다 낮기 때문)
* “mapping” : map-layer representation of state space!!
* A(grid 좌표 + corresponding value), device data, UAV flying times, UAV operational status
* 차원을 맞추면 M x M x n 차원의 tensor로 변환 가능
* “Map centering” : M x M x n -> (2M - 1) x (2M - 1) x n으로 Fig.2참조
* “Global-local map” : centered map을 가지고 local map + global map 2가지를 만듬 by local map function + global map function
* “Observation space”
* Local map + Global map + Flying time
* Observation space의 한 요소 = local observation of (agent I of the environment + data + remaining flying time + operational status) + global observation of (agent I of the environment + data + remaining flying time + operational status) + remaining flying time of agent I
* “multi-agent Q learning”
* Agent = homogeneous + noncommunicating.
* decentralized deployment(trained policy가지고 각 agent 별도 실시) or execution with centralized training(replay memory을 통해 training은 중앙에서 한 번에)
* 공통 reward X. Instead, individual but identical reward function

1. State

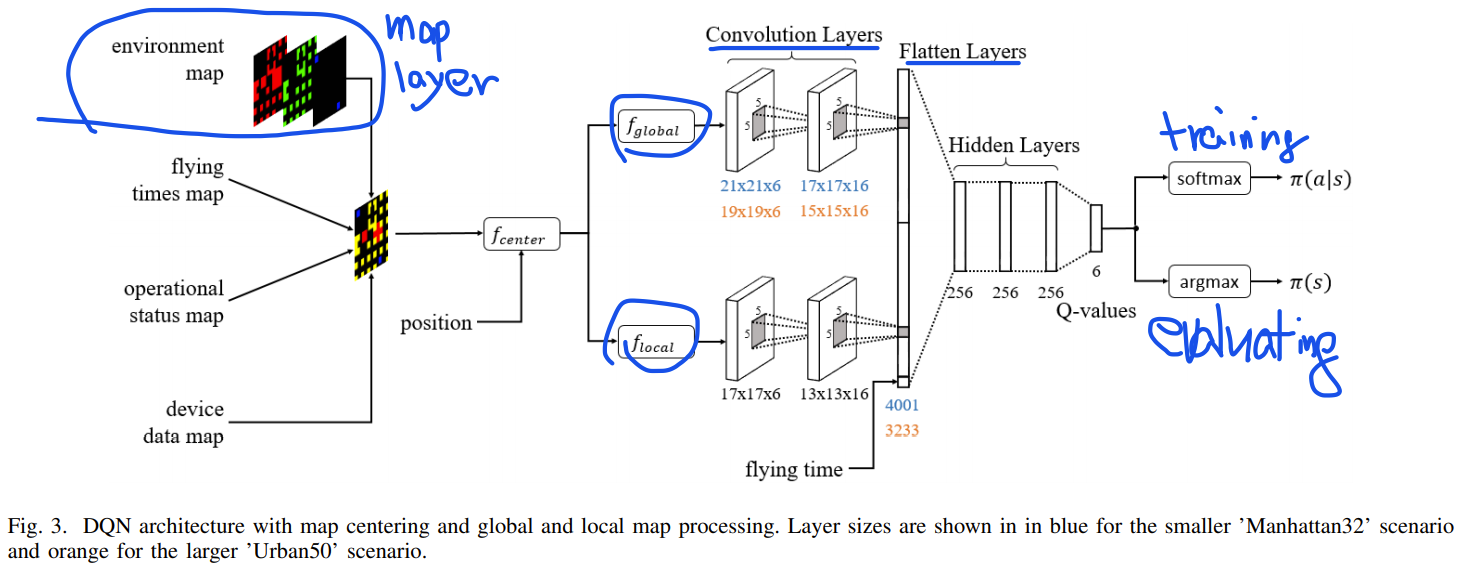
-environment information(Landing Zone + NFZs + Obstacles) + Agents(UAV Positions + Flying Times + Operational Status) + Devices(Device Positions + Device Data)

1. Action

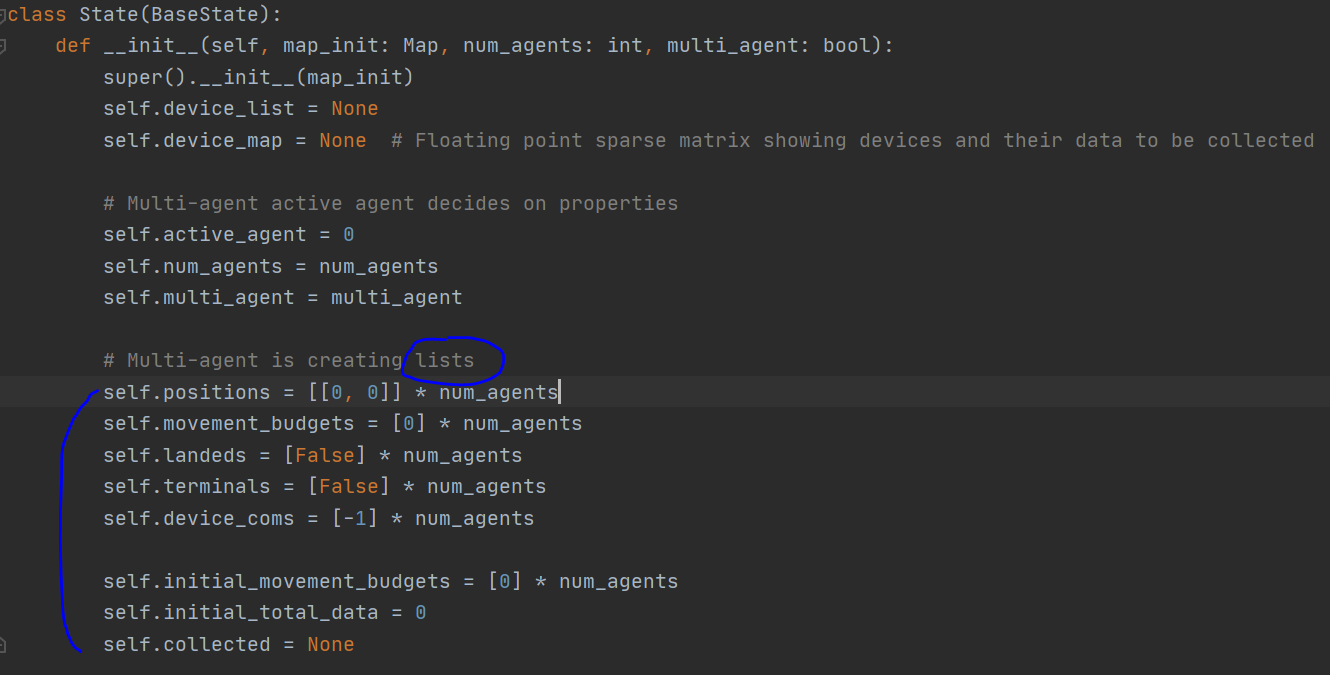
* Safety controller so as to collision avoidance + NFZ + obstacle avoidance + excluding landing + evaluates(해당 agent의 action을 수용할지 안할지, 안한다면 hovering)

1. Reward

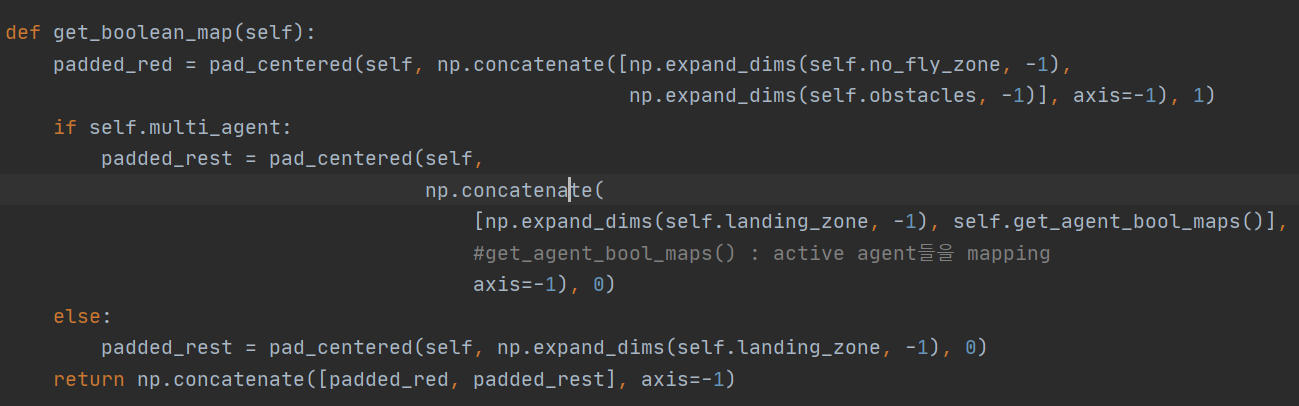
* Collective reward + individual penalty(when safety controller rejected) + individual penalty(when not landing) + constant movement penalty

1. Deep neural network
2. Code

* Multi-agent



마찬가지로, multi-agent을 위해 list 형태로 생성

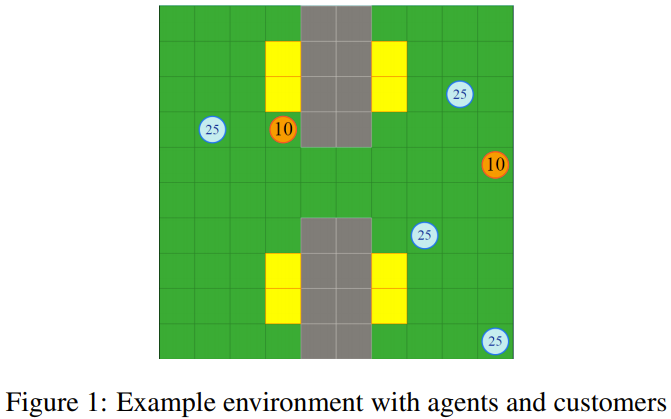
  
multi-agent(boolean)이면 여러 agent에 대한 좌표 mapping(get\_agent\_bool\_maps)을 추가로 concatenate

* Model : agent.py의 class DDQNAgent의 \_\_init\_\_ 부분 참고(논문에서 언급한대로 centralized training)

1. Autonomous Vehicle Fleet Coordination With Deep Reinforcement Learning

Cited by 2 times, ICLR 2018 Conference Blind Submission

1. 환경

* City dynamics that are associated with the Uber ride sharing platform

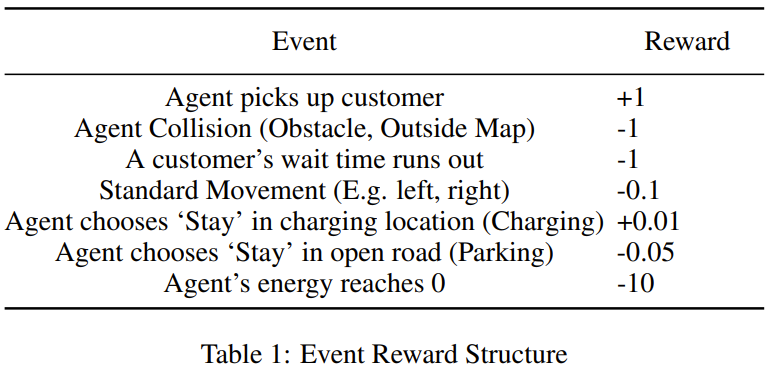
The general goal of each agent : travel to their customer and fulfill that demand.

Car(Blue circle) : agent, its actions are right, left, up, down, stay, has an energy level, agents can occupy same space without collision

Customer(Red circle) : goal location, ‘drop-off location’, ‘travel time’ determine how long agent is removed from the system while in transit.

Obstacles(Gray square) : locations that agents, customer can not travel to

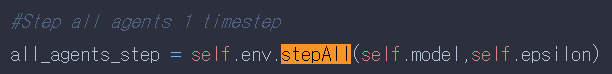
Charging stations(Yellow square) : agent refill their energy while in ‘stay’ action.

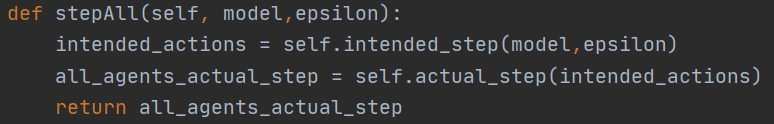
Open road(Green square)  


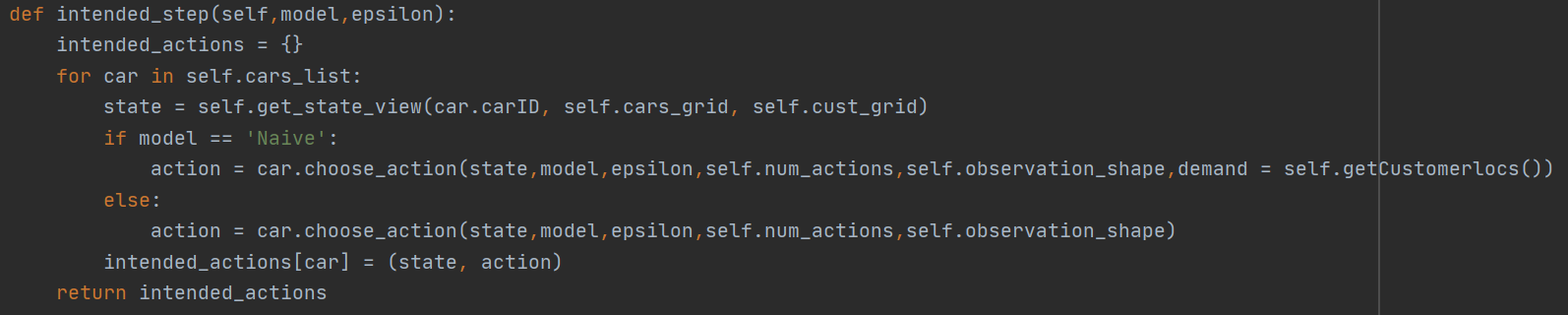
* Multi-agent 접근법 : centralized learning, decentralized learning 각각 장단점 있음 -> 절충안 = train and learn on the same network and share parameter + decentralized control -> each agents experiences are added to a common replay memory -> optimized when fitting a batch.
* Their respective view(1st layer) + partial observability -> independent behavior(next state를 모든 agent agents moved, all the conflict resolved 후의 상태로 표현했기 때문에 가능)
* PO-MADRL(Partially Observable Multi-Agent Deep RL) : each agent receive a private window, agent의 현재 위치에 따라 모든 방향으로 v space 즉, (2v + 1) 행렬을 가지게 함.
* Transfer learning : how to achieve multiple objects.(손님태우기 + battery level 유지) -> 손님태울때 reward는 주지만 energy 다 떨어지면 large negative reward 부여 -> 단순 reward structure optimal 도달 못함 -> transfer learning 도입(train PO-MADRL with an infinite amount of energy -> random energy level -> experience large negative reward -> can balance the two objectives)

1. State

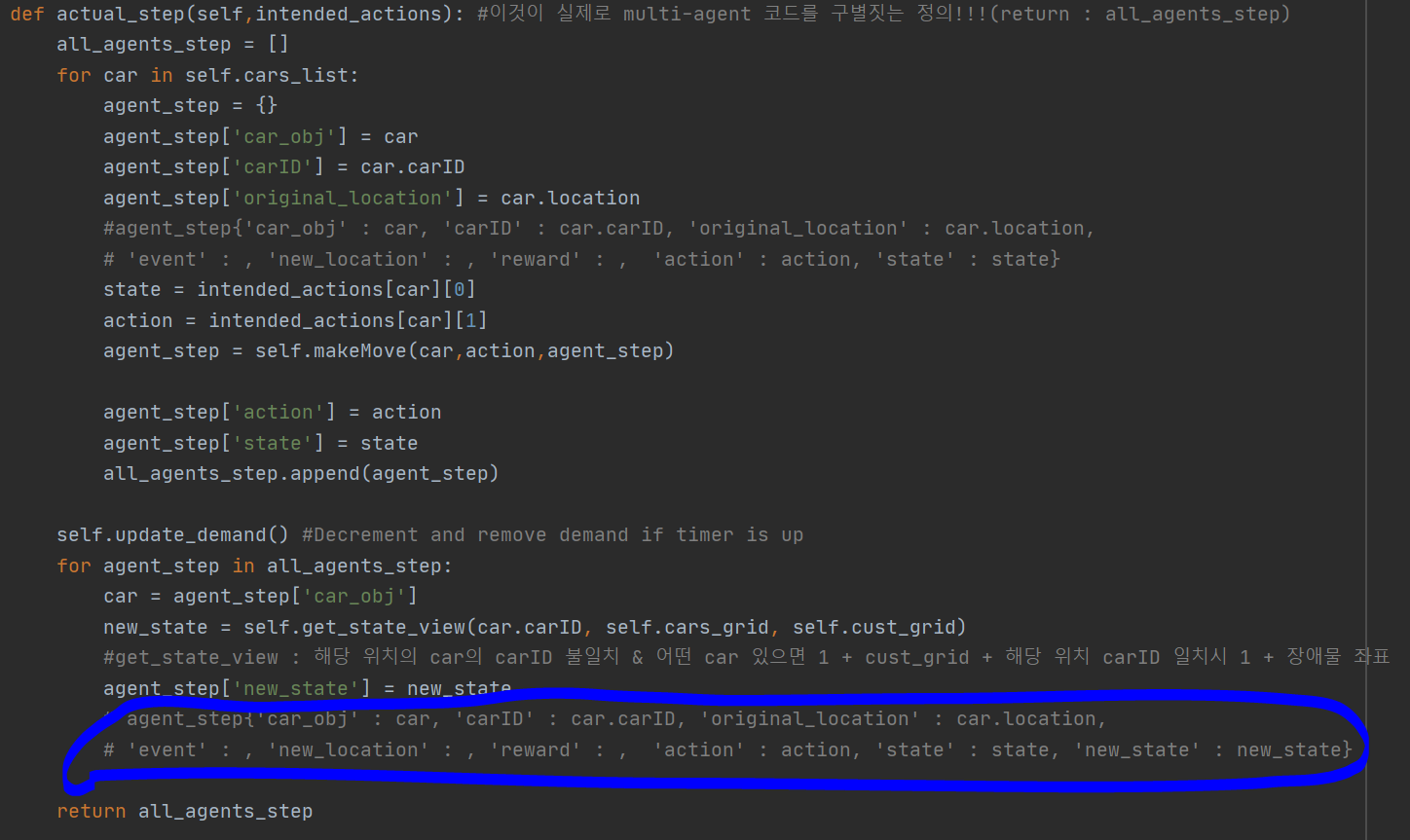
* 위에서 정의한 environment를 vector(image-like structure)로 표현 즉. 5 layers(CNN을 사용해야함)
* Self layer : 관심 agent 위치 인코딩(each self layer is unique to each agent)(value of 1 부여)
* Other agents layer : 관심 agent 이외의 agent 위치 인코딩(겹쳐있어도 1)
* Customer layer : all customer 위치 인코딩(value of wait time remained 부여)
* Obstacles layer : 1 = obstacles, 2 = charging station
* Extra agent layer : 관심 agent의 energy, priority 인코딩((0,0) = 남은 에너지, (0,1) = 우선 순위)

1. Code

Main 파일에서 multiagent과 관련있는 라인  




Intended\_step : Car list에 있는 car들의 state, action pair 반환

즉, for loop를 통해, 각 car에 대한 car obj, car id 등등의 dictionary를 리스트에 appending