Graph Convolutional Reinforcement Learning...

Seminarium: Reinforcement learning dla gier

Maria Wyrzykowska

https://arxiv.org/abs/1810.09202, Graph Convolutional Reinforcement Learning, Jiechuan Jiang, Chen Dun, Tiejun Huang, Zongqing Lu, ICLR'20

...czyli nauka kooperacji w systemach wieloagentowych

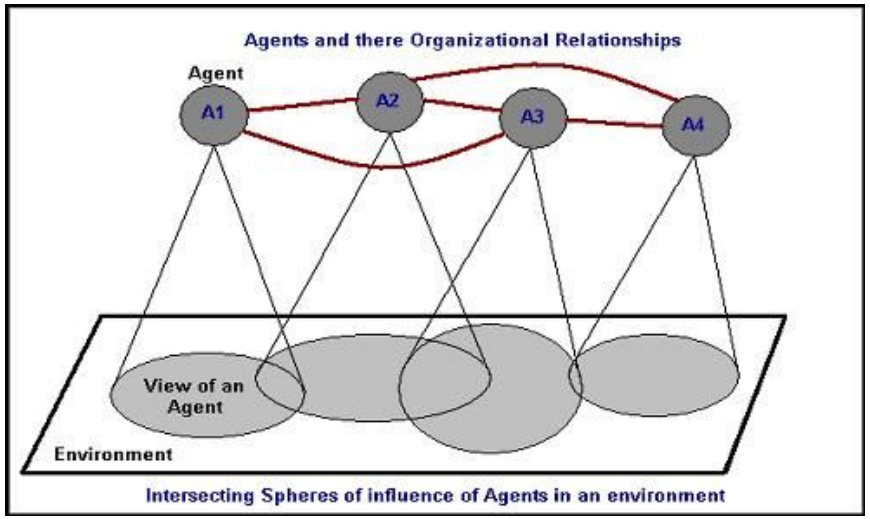
Seminarium: Reinforcement learning dla gier

Maria Wyrzykowska

https://arxiv.org/abs/1810.09202, Graph Convolutional Reinforcement Learning, Jiechuan Jiang, Chen Dun, Tiejun Huang, Zongqing Lu, ICLR'20

Systemy wieloagentowe (Multi Agent Systems)

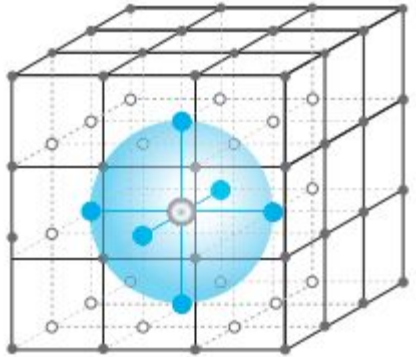
- wielu agentów, wiele obserwacji, dynamiczne środowisko
- zazwyczaj: agenci realizują wspólne cele
- kooperacja jest wartościowa
- przykłady:
  - sterowanie autonomicznymi pojazdami
  - kontrola sygnalizacji świetlnej
  - kontrola robotów (np eksploracja)



Różne podejścia do kooperacji: MeanField

Podejście 1:
"Standardowy" Q-learning

$$Q^j(s, \boldsymbol{a})$$



Problem:
- niepraktyczne dla dużej liczby agentów

Pomysł (Mean Field):
- aproksymujemy Q, uśredniając akcje sąsiadów danego agenta

$$Q^j(s, \boldsymbol{a}) = \frac{1}{N^j} \sum_k Q^j(s, a^j, a^k) \approx Q^j(s, a^j, \bar{a}^j) \qquad \bar{a}^j = \frac{1}{N^j} \sum_k a^k$$

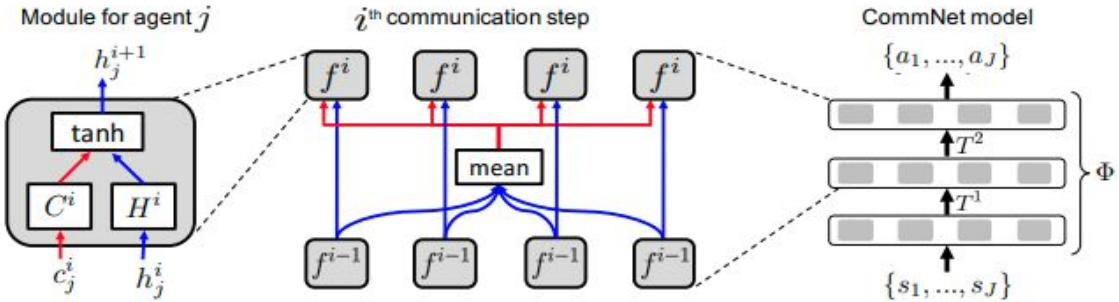
Problem: uśredniając, eliminujemy różnice między agentami, tracimy informacje

https://arxiv.org/pdf/1802.05438.pdf, Mean Field Multi-Agent Reinforcement Learning, Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, Jun Wang, ICML 2018

Różne podejścia do kooperacji: CommNet

Podejście 2:
Pozwalamy agentom komunikować się.

Przykładowa implementacja (CommNet):



Problem:

- ilu agentów powinno się ze sobą komunikować?

https://arxiv.org/pdf/1605.07736.pdf, Learning Multiagent Communication with Backpropagation, Sainbayar Sukhbaatar, Arthur Szlam, Rob Fergus, NIPS 2016

Różne podejścia do kooperacji: Casual Influence

Podejście 3:
Nagradzamy zachowania agentów, mające wpływ na innych.

Szczegóły:

Jak bardzo zmiana akcji agenta k zmieni dystrybucję prawdopodobieństw akcji agenta j?

$$c_t^k = \sum_{j=0, j\neq k}^{N} \left[D_{KL}[p(a_t^j \mid a_t^k, s_t^j) \Big\| \sum_{\tilde{a}_t^k} p(a_t^j \mid \tilde{a}_t^k, s_t^j) p(\tilde{a}_t^k \mid s_t^j)] \right]$$

Problem:

- Nagradzany wpływ na innych agentów może być negatywny

Rozwiązanie*: Wprowadzamy komunikację, nagradzamy wpływ komunikatów, a nie akcji

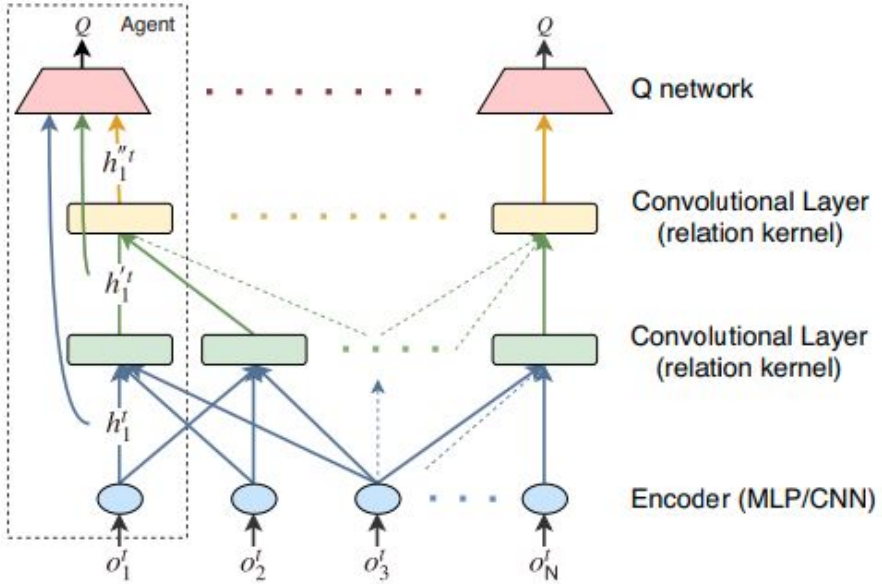
https://arxiv.org/abs/1810.08647, Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning, Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, Nando de Freitas, ICML 2018

Co chcemy osiągnąć?

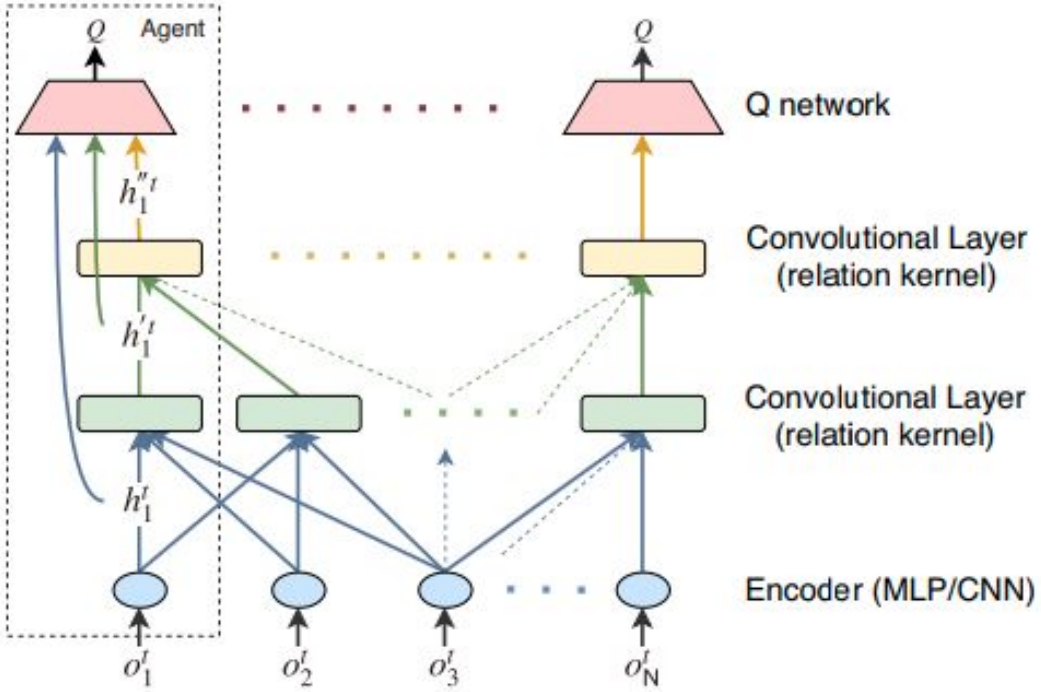
1. Komunikacja:
   a. szeroki zakres
   b. umiejętność decydowania, które informacje są ważne
2. Skalowalność do dużej liczby agentów
3. Consistent (spójna?) współpraca

Jak to chcemy osiągnąć?

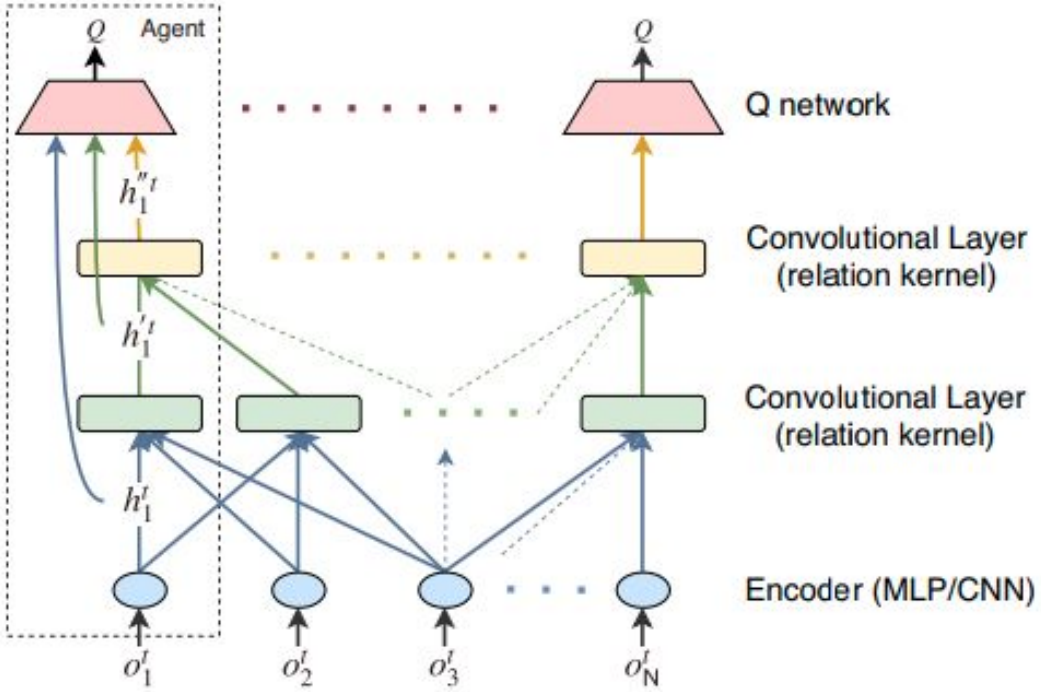
1. Komunikacja:
   a. szeroki zakres:
      GNN
   b. umiejętność decydowania, które informacje są ważne:
      attention (uwaga?)
2. Skalowalność do dużej liczby agentów:
   GNN, współdzielenie parametrów
3. Consistent (spójna?) współpraca:
   regularyzacja



Encoder

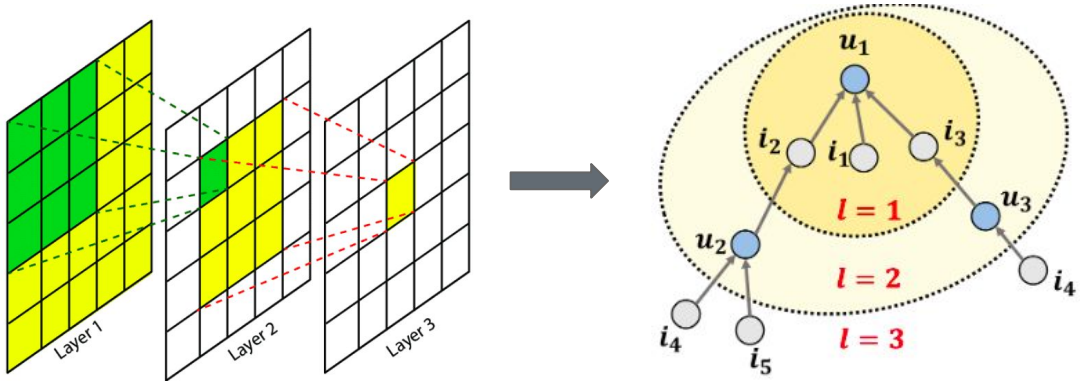

Q
Agent
Q
Q network
h''^t_1
Convolutional Layer
(relation kernel)
h'^t_1
Convolutional Layer
(relation kernel)
h^t_1
Encoder (MLP/CNN)
o^t_1
o^t_2
o^t_3
o^t_N

Konwolucje na grafach



Konwolucje na grafach

- zamiast sąsiednich pikseli, patrzymy na sąsiadów w grafie
- stała liczba parametrów
- kolejne warstwy = szersze pole widzenia
- detal implementacyjny: sąsiadów danego agenta kodujemy jako macierz, gdzie każdy wiersz to one-hot encoding indeksu kolejnego sąsiada

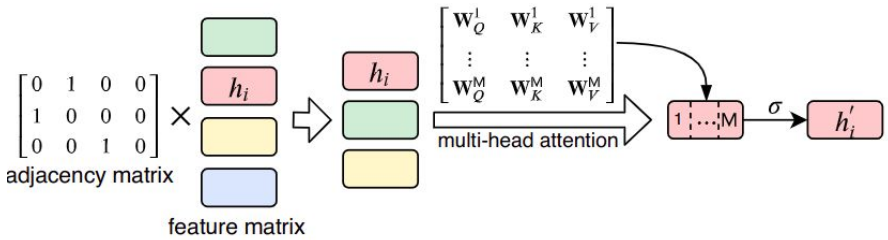


Konwolucje na grafach: kernel

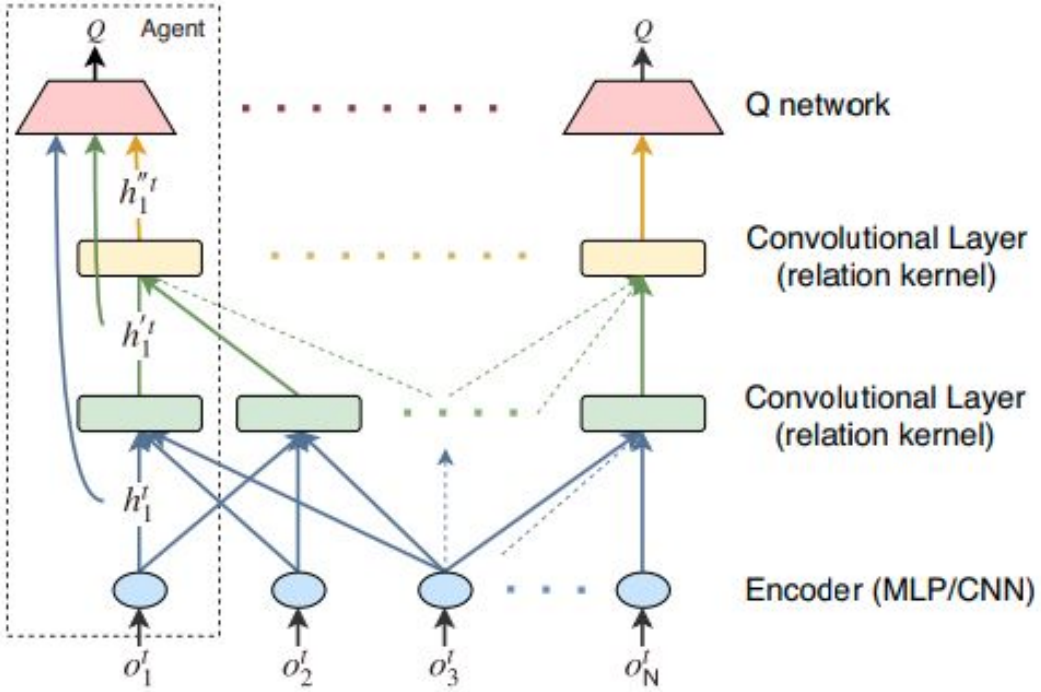
- pożądana cecha: niezależność od kolejności w której są sąsiedzi
- możliwości: uśrednianie, suma, ..., attention
- dla agenta *i* i jego sąsiada *j* oraz głowicy *m*:

$$\alpha_{ij}^{m} = \frac{\exp\left(\tau \cdot \mathbf{W}_Q^m h_i \cdot (\mathbf{W}_K^m h_j)^\mathsf{T}\right)}{\sum_{k \in \mathbb{B}_{+i}} \exp\left(\tau \cdot \mathbf{W}_Q^m h_i \cdot (\mathbf{W}_K^m h_k)^\mathsf{T}\right)}$$

$$h_i^{'} = \sigma(\text{concatenate}[\sum_{j \in \mathbb{B}_{+i}} \alpha_{ij}^{m} \mathbf{W}_V^m h_j, \forall m \in \mathsf{M}])$$



Q-learning


Q
Agent
Q
Q network
Convolutional Layer
(relation kernel)
Convolutional Layer
(relation kernel)
Encoder (MLP/CNN)

Q-learning

- input: konkatenacja outputów wszystkich poprzednich warstw
- gromadzimy memory buffer $(\mathcal{O}, \mathcal{A}, \mathcal{O}', \mathcal{R}, \mathcal{C})$ gdzie:
  - O - obserwacje (wszystkich agentów)
  - A - akcje
  - O’ - kolejne obserwacje
  - R - nagrody
  - C - macierze sąsiedztwa
- trening: standardowy*

$$\mathcal{L}(\theta) = \frac{1}{\mathsf{S}} \sum_{\mathsf{S}} \frac{1}{\mathsf{N}} \sum_{i=1}^{\mathsf{N}} \left(y_i - Q\left(O_{i,\mathcal{C}}, a_i; \theta\right)\right)^2$$

$$y_i = r_i + \gamma \max_{a'} Q\left(O'_{i,\mathcal{C}}, a'_i; \theta'\right)$$

- detal implementacyjny: graf dynamicznie się zmienia, ale macierze sąsiedztwa aktualizujemy w co drugim kroku

Regularyzacja

- **Cel:** kooperacja powinna być “stabilna”
- **Pomysł**: dystrybucje wag pochodzących z danego attention head dla danego agenta powinny być podobne w kolejnych momentach
- **Implementacja:** do funkcji starty dodajemy regularyzacje - karamy w zależności od tego, jak bardzo różnią się dystrybucje (KL)

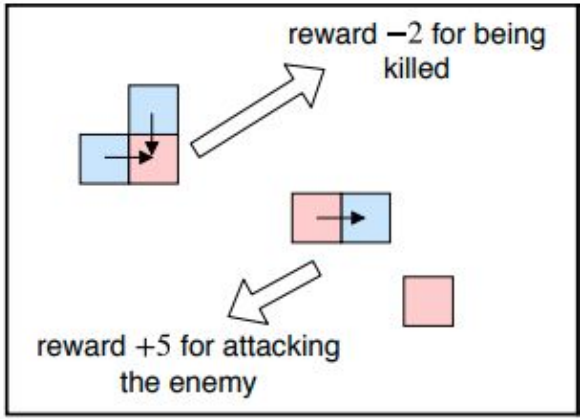
$$\mathcal{L}(\theta) = \frac{1}{\mathsf{S}} \sum_{\mathsf{S}} \frac{1}{\mathsf{N}} \sum_{i=1}^{\mathsf{N}} \left(\left(y_i - Q\left(O_{i,\mathcal{C}}, a_i; \theta \right) \right)^2 + \lambda \frac{1}{\mathsf{M}} \sum_{m=1}^{\mathsf{M}} D_{\mathrm{KL}}(\mathcal{G}_m^{\kappa}(O_{i,\mathcal{C}}; \theta) || \mathcal{G}_m^{\kappa}(O'_{i,\mathcal{C}}; \theta)) \right)$$

Eksperymenty

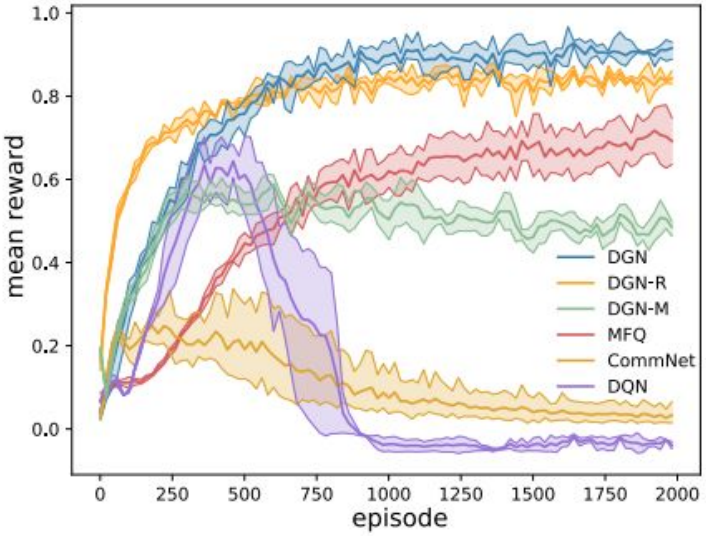
- platforma: MAgent
- plansza: 30x30, lokalne obserwacje agentów: 11x11
- scenariusze: battle, jungle, routing
- baseliny: independent Q-learning, DQN, CommNet, MeanField (MFQ)
- wersje DGN: DGN, DGN-R (bez regularyzacji), DGM-M (z średnią jako kernelem)
- wszystkie modele współdzielą parametry między agentami
- podobne rozmiary, takie same hiperparametry

Battle

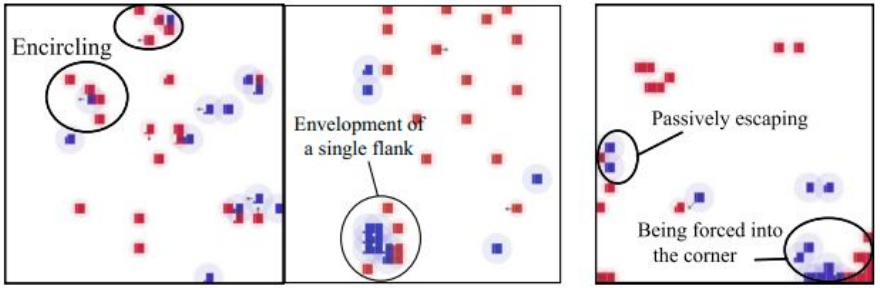
- 20 agentów vs 12 przeciwników (sterowanych przez DQN)
- agent: może się poruszać/atakować 4 sąsiednie pola
- przeciwnik: może się poruszać na 12 sąsiednich pól, atakować 8
- gracze mają po 6 hp, po śmierci są losowo respawnowani
- przeciwnicy są silniejsi niż agenci - by wygrać, muszą kooperować



Battle: wyniki



	DGN	DGN-R	DGN-M	MFQ	CommNet	DQN
mean reward	**0.91**	0.84	0.50	0.70	0.03	−0.03
# kills	**220**	208	121	193	7	2
# deaths	**97**	101	84	92	27	74
kill-death ratio	**2.27**	2.06	1.44	2.09	0.26	0.03

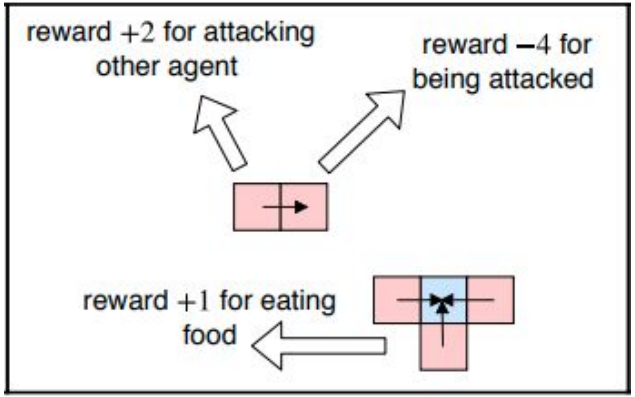


(a) DGN in battle

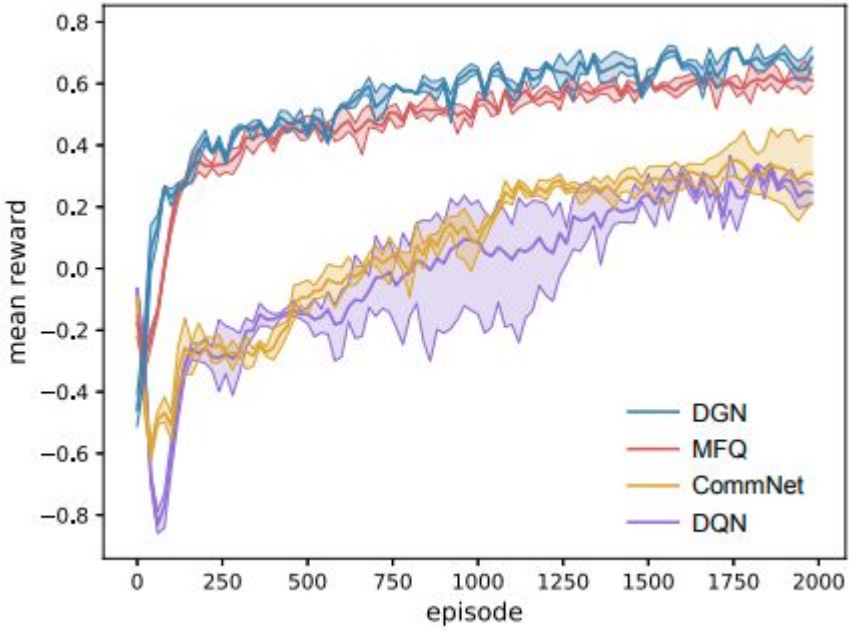
(b) DQN in battle

Jungle

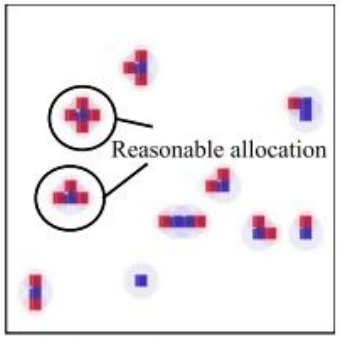
- 20 agentów, 12 jedzenia (stacjonarnego)
- agent: może się poruszać/atakować 4 sąsiednie pola
- nagroda za zaatakowanie agenta jest wyższa, niż za jedzenie
- agenci powinni nauczyć się dzielić zasobami, a nie atakować się wzajemnie



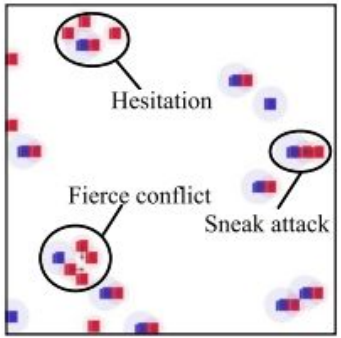
Jungle: wyniki



	DGN	MFQ	CommNet	DQN
mean reward	**0.66**	0.62	0.30	0.24
# attacks	**1.14**	2.74	5.44	7.35



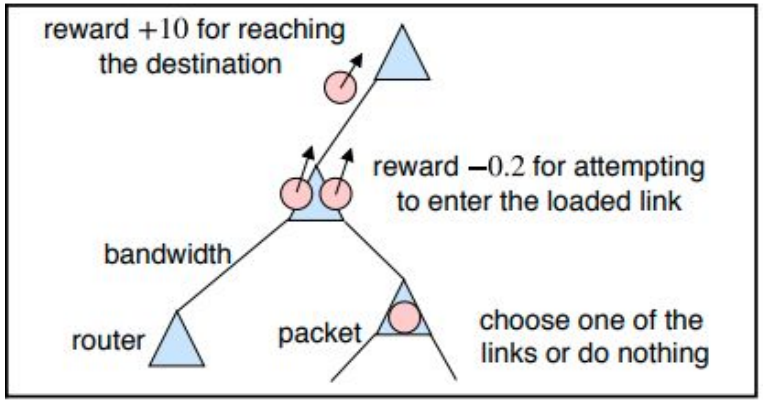
(c) DGN in jungle



(d) DQN in jungle

Routing

- 20 agentów-paczek z danymi (losowych rozmiarów), 20 routerów, każdy z nich połączony z 3 innymi
- każde połączenie ma jakiś bandwidth
- każda paczka ma source i destination
- obserwacje to: atrybuty paczki, atrybuty sąsiednich paczek i połączeń
- akcje to wybór kolejnych połączeń
- przejście paczki przez połączenie zajmuje czas liniowy do jego długości



Routing: wyniki

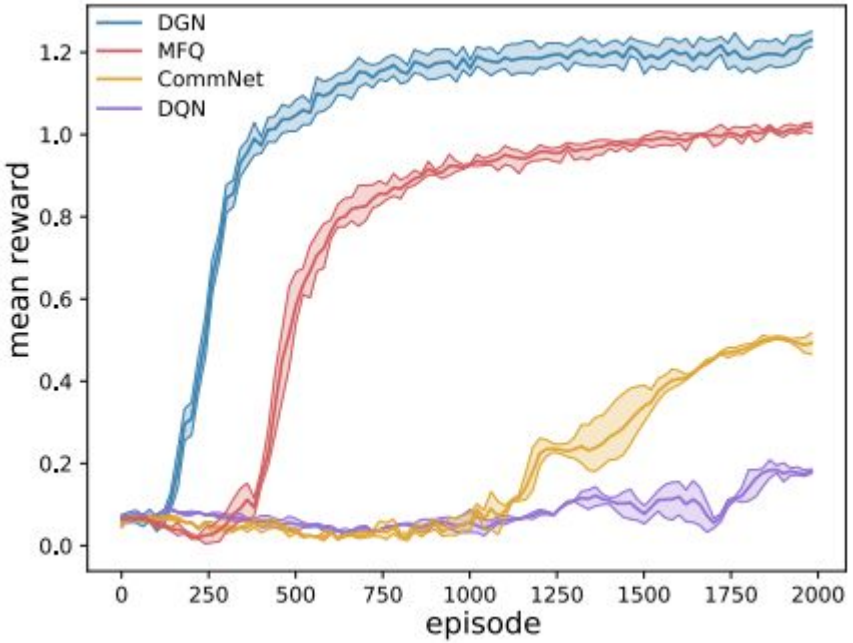
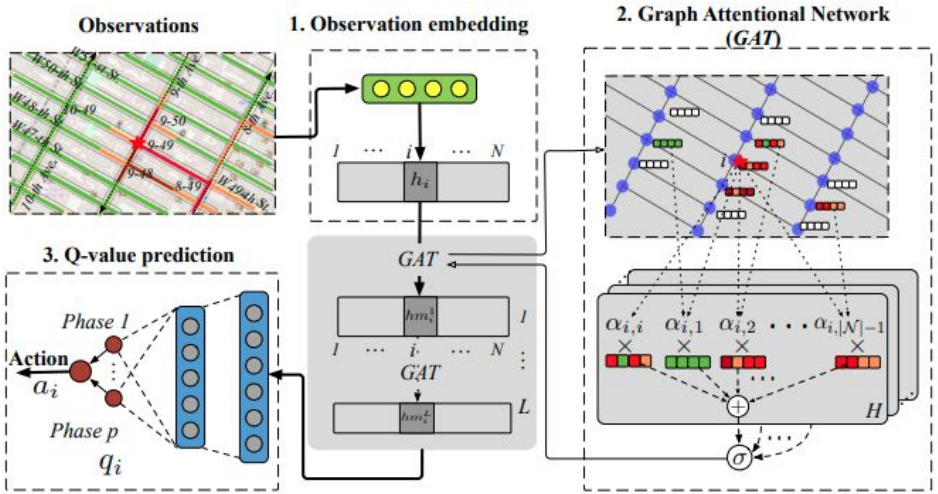


Table 3: Routing

(N, L)		Floyd	Floyd w/ BL	DGN	MFQ	CommNet	DQN
(20, 20)	mean reward			**1.23**	1.02	0.49	0.18
	delay	6.3	8.7	**8.0**	9.4	18.6	46.7
	throughput	3.17	2.30	**2.50**	2.13	1.08	0.43
(40, 20)	mean reward			**0.86**	0.78	0.39	0.12
	delay	6.3	13.7	**9.8**	11.8	23.5	83.6
	throughput	6.34	2.91	**4.08**	3.39	1.70	0.49
(60, 20)	mean reward			**0.73**	0.59	0.31	0.06
	delay	6.3	14.7	**12.6**	15.5	27.0	132.0
	throughput	9.52	4.08	**4.76**	3.87	2.22	0.45

delay: czas, jaki zajęła paczce podróż
throughput: liczba dostarczonych paczek/timestep

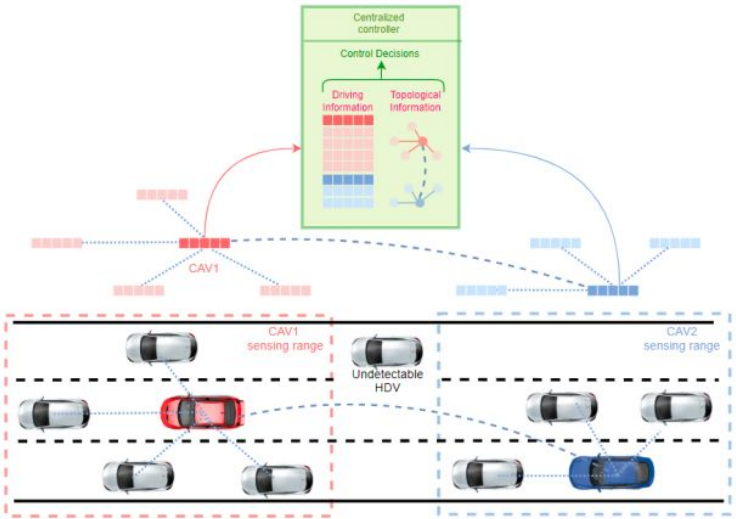
Zastosowania DGN: CoLight

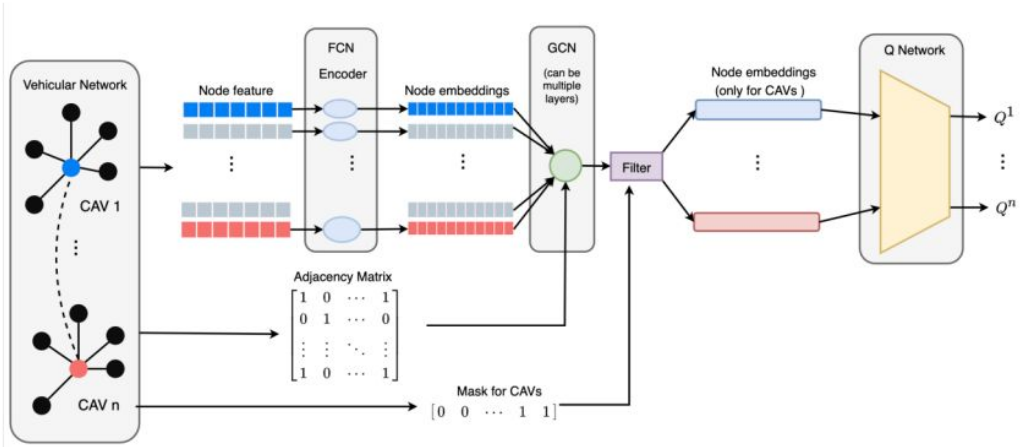


Model	$Grid_{6\times6}$-Uni	$Grid_{6\times6}$-Bi	$D_{NewYork}$	$D_{Hangzhou}$
Fixedtime [15]	209.68	209.68	1950.27	728.79
MaxPressure [24]	186.07	194.96	1633.41	422.15
CGRL [23]	1532.75	2884.23	2187.12	1582.26
Individual RL [30]	314.82	261.60	-*	345.00
OneModel [5]	181.81	242.63	1973.11	394.56
Neighbor RL [1]	240.68	248.11	2280.92	1053.45
GCN [18]	205.40	272.14	1876.37	768.43
CoLight-node	178.42	176.71	1493.37	331.50
CoLight	**173.79**	**170.11**	**1459.28**	**297.26**

https://arxiv.org/abs/1905.05717, CoLight: Learning Network-level Cooperation for Traffic Signal Control, Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, Zhenhui Li, ACM, 2018

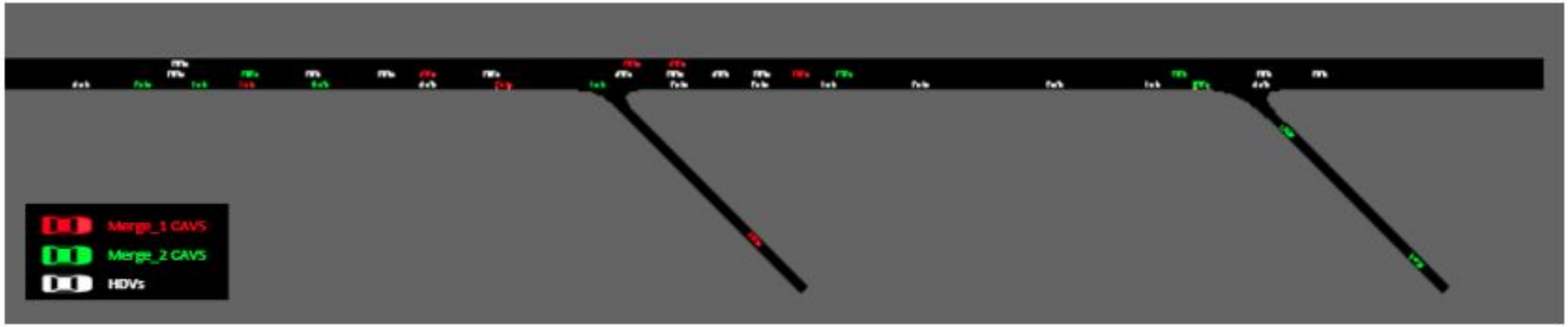
Zastosowania DGN: CAV

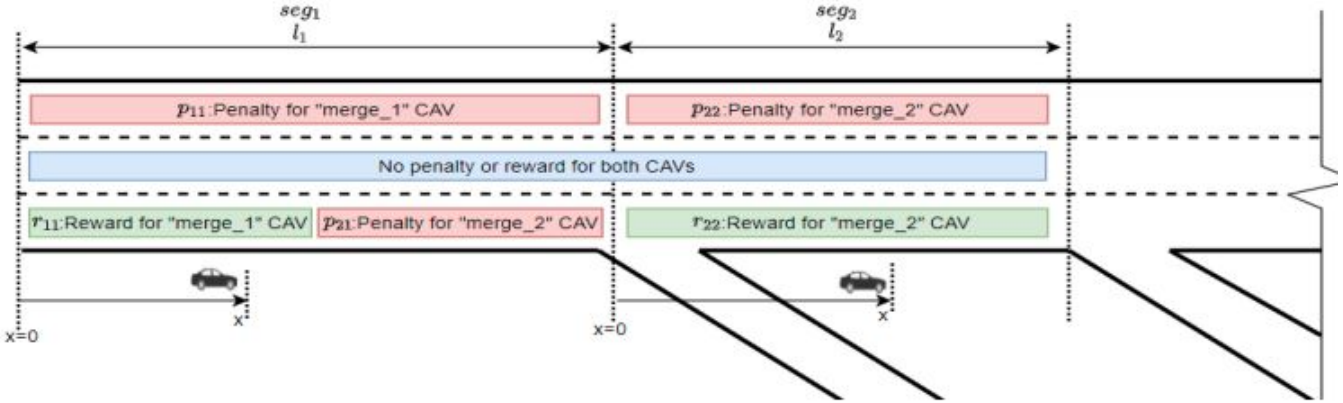




https://arxiv.org/abs/2010.05437, A DRL-based Multiagent Cooperative Control Framework for CAV Networks: a Graphic Convolution Q Network, Jiqian Dong, Sikai Chen, Paul Young Joun Ha, Yujie Li, Samuel Labi, TRB 2021

Zastosowania DGN: CAV





Wnioski

- MARL jest ciekawą dziedziną, stawiającą wiele wyzwań i mającą wiele aplikacji w życiu
- Zastosowanie attention (jak zwykle?) wydaje się być wartościowe
- Praca o DGN była trochę krytykowana jako niejasna, ale autorzy wkładają wysiłek, żeby ją poprawiać
- Wiarygodność wyników z pracy o DGN może nie jest bardzo duża, ale zastosowanie architektury w innych pracach trochę bardziej do nich przekonuje

Dziękuję za uwagę!