行動決定優先度を考慮したマルチエージェントの系列的協調行動生成

○山下 佳威, 小松 廉, 永谷 圭司, 淺間 一, 安 琪, 山下 淳 (東京大学)

Sequential Cooperative Action Generation in Multi-Agent Systems Considering Action Decision Priorities

○ Kai Yamashita, Ren Komatsu, Keiji Nagatani, Hajime Asama, Qi An, Atsushi Yamashita

(The University of Tokyo)

Abstract: This research deals with cooperative action generation using deep learning in multi-agent systems. In order to generate cooperative actions, it is necessary to take into account the actions of other agents that have already made decisions, rather than allowing each agent to decide its own actions independently. Based on the assumption that the priority of action decisions is important to generate cooperation, which is a mutual relationship among agents, we proposed a network to determine the order of action decisions. By combining the proposed method with the Transformer-based previous method for sequentially determining the actions of multiple agents, we verified the effectiveness of the proposed method on the SMACv2 benchmark.

1. 序論

マルチエージェントシステムは, 複数のエージェントからなるシステムであり，それぞれのエージェントが同時に異なる動作を実行可能であることから, 大規模な問題を効率的に解決できる可能性がある．また，一部のエージェントが停止した場合においても直ちにシステム全体が停止しないといった頑健性も有していることから, 近年注目を集めている [1].

しかし, マルチエージェントシステムが置かれる行動空間は膨大かつ複雑であり, それらに対して人間が最適な行動を記述するのは困難である. したがって , マルチエージェントシステムに属するエージェントが自ら行動を学習し, 最適な行動を獲得することが望ましい.

マルチエージェントシステムが対象とするタスクには, エージェントが協調してタスク達成を行うものがある．このような協調タスクにおいては，個々が独立に行動を決定するのではなく，すでに決定した他のエージェントの行動を考慮にいれることが重要である．例えば，エージェント間でオブジェクトを受け渡しながら，あるゴール地点までそのオブジェクトを運搬するようなタスクでは，オブジェクトを受け取るエージェントはオブジェクトを把持しているエージェントの計画する移動方向などを鑑みた経路計画をすることによって効率的な動作を行うことが可能となる．また，他のエージェントの行動を考慮する際には，エージェントがどの順番で行動を決定するのかが重要である．

そこで，本研究では行動決定順番を決定するネットワークを提案し．本提案手法を先行研究である Transformer を用いた複数エージェントの行動を系列的に決定する手法 [2] に組み合わせることで，SMACv2 という協調的マルチエージェント環境において本提案手法の有効性を確認する.

2. 先行研究

2.1 マルチエージェント強化学習

マルチエージェント強化学習の手法は TRPO [3], PPO [4] のような単一のエージェントにおける強化学習の手法を拡張した手法が多数開発されている．代表例として, MAPPO [5] はそのような手法の一つであり，HAPPO [6] は MAPPO をさらに発展させ, 複数種類のエージェントから構成される協調的マルチエージェントシステムにおいて優れた性能を示した.

2.2 系列モデルを用いたマルチエージェント深層強化学習

マルチエージェント強化学習において，行動生成問題を系列行動生成モデルとして扱ったのが Wen らによる Multi Agent Trasformer（MAT）である [2]．系列情報を扱うネットワークとして GRU [7] や LSTM [8] のような RNN の他に，Vaswani らによって提唱された Transformer [9] がある．Attention 機構からなる Transformer は文章生成などの自然言語処理タスクで極めて優れた性能を示している．また，Transformer は自然言語処理タスクのみならず，Vision Transformer [10] といった画像認識モデルや DEtection Transformer [11] のような物体検出モデル，シングルエージェントの強化学習モデルである Decision Transformer [12] など，Transformer をベースとしたモデルが各分野で提案されている.

MAT では，Transformer の Encoder に各エージェントの観測系列を入力し, Decoder では既に決定した m 番目までのエージェントの行動と, Encoder で変換した観測系列をもとに $m+1$ 番目のエージェントの行動を決定するといったように，既に決定したエージェントの行動を考慮しながら行動決定を行うという自己回帰的な系列的行動生成により，様々な協調的マルチエージェント環境において, 非常に高い性能を示した.

しかし, MAT はエージェントの行動を系列的に決定するが, 行動決定のエージェントの順番は予め固定されるか, ランダムに生成される．系列的な決定では, 行動決定の順番が遅いエージェントほど他のエージェントの行動を考慮した行動を生成する. したがって , エージェントの行動決定順番は状況に応じて決定されることが望ましい.

2.3 組み合わせ最適化

最適な順列を解とする問題の一つとして，巡回セールスマン問題のような組み合わせ最適化問題があり，これを深層学習を用いた手法によって解く研究が多数行われている．Pointer Network [13] は順列の学習を

行うために，再帰的ニューラルネットワークを用いて各要素の状態 $\boldsymbol{o} := o_1, o_2, \cdots$ をもとに要素の順列 $\boldsymbol{i} = i_1, i_2, i_3, \cdots$ を出力し，巡回セールスマン問題や最小カット問題などの組み合わせ最適化問題を解く手法を提案した．

また，深層強化学習を用いて組み合わせ最適化問題を解く研究も多数存在し，Kool らは再帰的ニューラルネットワークではなく，Transformer の Encoder と Attention 機構を用いることによって巡回セールスマン問題等を解く手法を提唱した [14]．Graph Attention Network をもちいる Lu [15] らの研究や，LSTM を用いる Chen らの研究 [16] も深層強化学習を用いた最適化を行っている．

本研究では，エージェントの系列的行動生成における行動決定順番を組み合わせ最適化問題を解く深層強化学習手法と組み合わせることによって行動決定順番を考慮することが可能になるという仮定のもと，MAT と行動決定順番を決定する深層強化学習手法により性能の改善を目指す．

3. 提案手法

3.1 マルコフ決定過程

協調的なマルチエージェントシステムでは，マルコフ問題として問題を定義することができる．すなわち，$< \mathcal{N}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{A}}, R, P, \gamma >$ として定式化することができる [17]．ただし，$\mathcal{N} = \{1, 2, \cdots, N\}$ はエージェントの集合，$\boldsymbol{\mathcal{O}} = \prod_{i \in \mathcal{N}} \mathcal{O}^i \subset \mathbb{R}^{N \times D}$ はエージェントの観測集合の直積（D は次元数），$\boldsymbol{\mathcal{A}} = \prod_{i \in \mathcal{N}} \mathcal{A}^i$ はエージェントの行動集合の直積，$R : \boldsymbol{\mathcal{O}} \times \boldsymbol{\mathcal{A}} \longrightarrow \mathbb{R}$ は報酬関数であり，$P : \boldsymbol{\mathcal{O}} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{O}} \longrightarrow \mathbb{R}$ は状態遷移関数で，$\gamma \in (0, 1)$ は割引率である．

このとき，学習すべき行動は $\pi_i : \boldsymbol{\mathcal{O}} \times \mathcal{A}^i \longrightarrow [0, 1]$ と表現されるエージェント群の観測をもとに行動を決定するマルコフ定常方策である．この際に割引和報酬 $\Sigma_{t=0}^{\infty} \gamma^{t-1} r_t$ を最大化する方策を学習する．

$$Q_\pi(\boldsymbol{o}, \boldsymbol{a}) = \mathbb{E}_{o_{1:\infty} \sim P, a_{1:\infty} \sim \pi}[\Sigma_{t=0}^{\infty} \gamma^{t-1} r_t | \boldsymbol{o_0} = \boldsymbol{o}, \boldsymbol{a_0} = \boldsymbol{a}], \tag{1}$$

$$V_\pi(\boldsymbol{o}) = \mathbb{E}_{o_{1:\infty} \sim P, a_{0:\infty} \sim \pi}[\Sigma_{t=0}^{\infty} \gamma^{t-1} r_t | \boldsymbol{o_0} = \boldsymbol{o}], \tag{2}$$

のように状態価値関数，行動価値関数を定義することができる．ここで，t はタイムステップを表し，$\boldsymbol{o}_t \in \boldsymbol{\mathcal{O}}, \boldsymbol{a}_t \in \boldsymbol{\mathcal{A}}$ は時間 t のときのエージェント群の観測と行動を表す．状態価値関数は初期状態 $t = 0$ の時の観測が条件づけられているとき，行動価値は初期状態のときの観測と行動が条件づけられている際，エージェント群が方策群 $\boldsymbol{\pi} = \prod_{i \in \mathcal{N}} \pi_i$ に従って行動する際の割引報酬和の期待値である．そして，Advantage 関数を $A_\pi(\boldsymbol{o}, \boldsymbol{a}) := Q_\pi(\boldsymbol{o}, \boldsymbol{a}) - V_\pi(\boldsymbol{o})$ のように定義する．

3.2 エージェントの系列的行動生成

本提案手法においては，Wen らによる MAT [2] を系列的行動生成を行うベースモデルとして用いる．本手法の概要を図 1 に示す．本モデルは，大別して Encoder, Decoder, Order Network という 3 つのコンポーネントから構成される．エージェント群の観測 $\boldsymbol{o} \in \boldsymbol{\mathcal{O}}$ は Transformer Encoder により構成される Encoder を経由することにより出力として $\hat{\boldsymbol{o}} \in \mathbb{R}^{N \times D}$ を得る．さらに，全結合ニューラルネットワークを経由して状態価値 $V(\hat{\boldsymbol{o}}) \in \mathbb{R}$ を計算する．

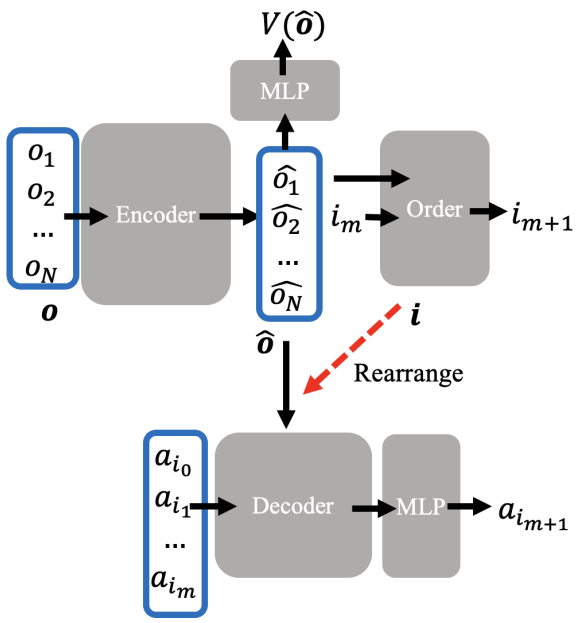

Fig.1 Overview of the Proposed Method. The proposed model consists of the Encoder, Decoder, and Order Network components: the Order Network outputs the order in which the agents decide their actions, and the Decoder decides the actions of each agent based on the order and the output of the Encoder.

そして，Transformer Decoder と全結合ニューラルネットワークより構成される Decoder では $\hat{\boldsymbol{o}}$ を Order Network より出力されたエージェントの順番 $\boldsymbol{i}$ によって並び替えたものと，既に決定したエージェントの行動 $a_{1:m}$ を用いて次のエージェントの行動の確率 $\pi(a_{m+1} | \hat{\boldsymbol{o}}, \boldsymbol{a}_{1:m})$ を出力し，行動を決定する．

3.3 行動決定順番の優先度決定手法

行動決定順番を出力するため，Transformer の Multi Head Attention 機構を用いたニューラルネットワークから構成される Order Network というコンポーネントを提案する．MAT の Encoder の出力である $\hat{\boldsymbol{o}}$ を用いて，自己回帰的に行動決定順 $\boldsymbol{i} = i_1, i_2, i_3, \cdots$ を決定する．

m 番目までの行動決定順が既に計算されていると仮定する．

系列の位置をベクトル表現するために，下の関数を用いる．

$$f(p, 2d) = \sin(\frac{p}{10000^{2d/D}}), \tag{3}$$

$$f(p, 2d+1) = \cos(\frac{p}{10000^{2d/D}}). \tag{4}$$

ここで，$f(\cdot, \cdot) \in \mathbb{R}^{N \times D}$ は $f(x, y)$ において $x \in \mathcal{N}$ が系列内の位置，$1 \leq y \leq D$ は各ベクトルの次元方向のインデックスである．

$m+1$ 番目に行動を決定するエージェントを決定するために，以下のクエリベクトル $q_m \in \mathbb{R}^{3D}$ を考える．

$$q_m = \mathrm{Concat}(\bar{o}, \hat{o}_{i_m}, f(m, \cdot)), \tag{5}$$

となる．ここで，$\bar{o} = \frac{1}{N} \sum_i \hat{\boldsymbol{o}}_i$ で，Encoder の出力の $\hat{\boldsymbol{o}}$

を平均したものである．このベクトルと，直前に選択されたエージェント i_m の $\hat{o}_{i_m}$ と，m 番目の位置表現ベクトル $f(m,\cdot) \in \mathbb{R}^D$ を連結してクエリベクトルを作成し，これを用いて $\hat{q_m} \in \mathbb{R}^D$ を Multi Head Attention 機構の出力として得る．

$$\hat{q}_m = A_{\text{multi}}(Q = q_m, K = \hat{\boldsymbol{o}}, V = \hat{\boldsymbol{o}}, \text{mask} = M^m), \quad (6)$$

$$M^m \in \{0,1\}^N, M^m_j = \mathbb{I}_{j \in \boldsymbol{i}_{1:m}}. \quad (7)$$

ただし，$A_{\text{multi}}(Q = \cdot, K = \cdot, V = \cdot, \text{mask} = \cdot)$ は Multi Head Attention 機構である．Attention 機構を以下に示す．

$$A(Q,K,V,\text{mask} = M) = \text{softmax}((\frac{Q^T K}{\sqrt{D}}) - M \cdot \infty)V, \quad (8)$$

$$A_{\text{multi}}(Q,K,V,\text{mask} = M) = \text{Concat}(Z_1, Z_2, \cdots, Z_h)W_o, \quad (9)$$

$$Z_i = A(W_i^Q Q, W_i^K K, W_i^V V, \text{mask} = M). \quad (10)$$

ここで，W_i^Q, W_i^K, W_i^V, W_o は変換行列であり，学習パラメータである．Attention 機構は Query（Q）と Key（K）の内積の値で Value(V) の重み付け和をとる．そして，単一の Attention 機構の出力を並列化し，連結したのち変換したものが Multi Head Attention である．マスク M^m によって, Attention 機構において既に選択されたエージェントの情報は加算されない．

$\hat{q}_m$ を用いて m+1 番目に行動を決定するエージェントの確率分布を得る．

$$\pi_i(i_{m+1}|\hat{\boldsymbol{o}}, \boldsymbol{i}_{1:m}) = \text{softmax}((\frac{W_q \hat{q}_m \cdot W_k \hat{\boldsymbol{o}}}{\sqrt{D}}) - M^m \cdot \infty). \quad (11)$$

マスク M^m によって，既に選択されたエージェントの確率は 0 となる．ここで，$W_q, W_k \in \mathbb{R}^{D \times D}$ と開始トークン $\hat{o}_{i_0}$ は学習するモデルパラメータである．

このようにして，Order Network においてエージェントの行動決定順を決定し，出力されたエージェントの行動決定順番 $\boldsymbol{i}$ をもとに，並び替えられた状態系列 $\hat{\boldsymbol{o}}_{\boldsymbol{i_{1:N}}} = \hat{o}_{i_1}, \hat{o}_{i_2}, \cdots \hat{o}_{i_N}$ と Decoder によって各エージェントの行動を決定する．

3.4 学習

学習を行うに際して，損失関数を定義する. モデル全体のパラメータを θ とし，最適化には以下の損失関数を用いる．

$$L_{\text{Encoder}}(\theta) = \frac{1}{TN} \sum_{m \in \mathcal{N}} \sum_t \left[r(\boldsymbol{o_t}, \boldsymbol{a_t}) + \gamma V_{\theta^-}(\hat{\boldsymbol{o}}^{\boldsymbol{i_m}}_{\boldsymbol{t+1}}) - V_\theta(\hat{\boldsymbol{o}}^{\boldsymbol{i_m}}_{\boldsymbol{t}}) \right]^2, \quad (12)$$

$$L_{\text{Decoder}}(\theta) = -\frac{1}{TN} \sum_{m \in \mathcal{N}} \sum_t \min(r_t^{i_m}(\theta)\hat{A}_t, \text{clip}(r_t^{i_m}(\theta), 1 \pm \epsilon)\hat{A}_t), \quad (13)$$

$$r_t^{i_m}(\theta) = \frac{\pi_\theta^{i_m}(a_t^{i_m}|\hat{\boldsymbol{o}}_{\boldsymbol{t}}, \boldsymbol{i}, \hat{\boldsymbol{a}}_{\boldsymbol{t}}^{\boldsymbol{i_{1:m-1}}})}{\pi_{\theta_{\text{old}}}^{i_m}(a_t^{i_m}|\hat{\boldsymbol{o}}_{\boldsymbol{t}}, \boldsymbol{i}, \hat{\boldsymbol{a}}_{\boldsymbol{t}}^{\boldsymbol{i_{1:m-1}}})}, \quad (14)$$

$$L_{\text{Order}}(\theta) = -\frac{1}{T} \sum_t \min(r_t^i(\phi)\hat{A}_t, \text{clip}(r_t^i(\phi), 1 \pm \epsilon)\hat{A}_t), \quad (15)$$

$$r_t^i(\theta) = \prod_{m \in \mathcal{N}} \frac{\pi_\theta(i_m^t|\hat{\boldsymbol{o}}_{\boldsymbol{t}}, \boldsymbol{i}_{0:m-1}^t)}{\pi_{\theta_{\text{old}}}(i_m^t|\hat{\boldsymbol{o}}_{\boldsymbol{t}}, \boldsymbol{i}_{0:m-1}^t)} \quad (16)$$

$$L(\theta) := L_{\text{Encoder}}(\theta) + L_{\text{Decoder}}(\theta) + L_{\text{Order}}(\theta). \quad (17)$$

ここで, T は episode length を表し, タイムステップ t について $0 \leq t \leq T-1$ である．Encoder では TD ターゲットとの誤差を最小化するように学習を行い，Decoder, Order Network の方では PPO の損失関数 [4] を最小化する．これら 3 つの損失関数を合計した $L(\theta)$ を最小化するよう，勾配降下法によって θ を最適化する．ただし，$\hat{A}_t$ は Advantage 関数であり，Encoder より出力される状態価値 $V_\theta(\hat{\boldsymbol{o}})$ と Generalized Advantage Estimation [18] によって計算される．また，θ^- は target network [19] である．

4. 実験

4.1 シミュレーション環境

協調的マルチエージェント強化学習の環境として様々な研究で幅広く用いられているものとして，Star Craft Multi Agent Challenge（SMAC）[20] がある．SMAC はリアルタイムストラテジーゲームである Star Craft Ⅱ を協調的マルチエージェント環境として提供するものであり，様々な先行研究による取り組みにより，MAT をはじめとした様々なモデルで SMAC 上ではほぼ性能の上限値を達成している．これは，エージェントのチーム編成やエージェントの配置等が固定されており，定常的な環境であることに起因する．

Ellis らによって開発された SMACv2 [21] は環境に非定常性を与えることによって，非定常なシナリオにもモデルが汎化できることを要求する．SMACv2 においては，環境開始時にエージェントの編成が一定の確率からランダムに編成され，また初期位置もランダムに配置される. SMACv2 には（a）Terran,（b）Protoss,（c）Zerg の三種類のシナリオが存在する．それぞれのシナリオで自チームのエージェント群は 3 種類のエージェントから一定確率で一台ずつエージェント群の台数が設定された台数になるまで決定される．SMACv2 においては，エージェント編成が複数の種類のユニットから構成されるため，単一のユニットから構成される環境に対し行動決定順序の影響がより大きいと考えられる．

本研究においては，この SMACv2 の Protoss シナリオにおいて，Wen らによる MAT と行動決定順番を決定する本提案手法それぞれにおいて学習を行い，一定学習ステップごとに 200 エピソード分評価実行を行い，

Table 1 hyper parameter config

hyper parameters	value
learning rate	5.0e-4
optimizer	Adam
optimizer eps	1.0e-5
hidden layer dim	64
num Attention Head	1
num Transformer Layer	1
episode length	100

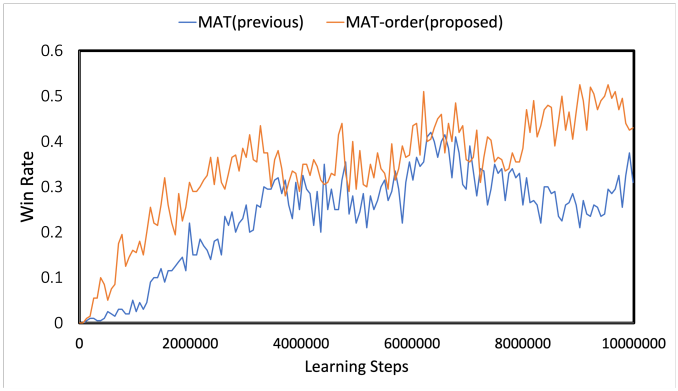

Fig.2 Experiment Results

200 エピソード間における勝率を測定し，比較することで性能の評価を行った．

4.2 学習設定

学習を行うにあたり，各種ハイパラメータの設定を表 1 に示す．実装は機械学習 Python ライブラリの PyTorch [22] を用いて行い，GPU に NVIDIA GeForce RTX 3090 を用いて学習を行った．

5. 結果

図 2 に，Protoss シナリオにおける，Wen らの MAT と行動決定順番を決定するニューラルネットワークを組み合わせた本提案手法を用いた比較実験の結果を示す．

Protoss シナリオにおいては，行動決定順番を決定する機構を導入した本提案手法が Wen らの MAT よりも高い勝率を達成していることが確認された．

6. 結論

本研究では，協調的マルチエージェントシステムにおいて，エージェントの行動決定優先度を考慮し，行動決定順番を決定する機構と，系列モデルを用いてエージェント群の行動を系列的に生成する先行研究の手法と組み合わせる手法を提案した．そして，協調的マルチエージェントシステム環境の一つである SMACv2 上において，比較実験を実施し，本提案手法の有効性を検討した結果，一部環境下においての有効性を確認した．

将来展望として，他の協調的マルチエージェント環境下においての性能評価，およびネットワークより出力される行動決定順番の詳細な分析を行う．

謝辞

本研究の一部は，JST【ムーンショット型研究開発事業】グラント番号【JPMJMS2032】の支援を受けたものである．

参考文献

[1] 淺間一: “マルチエージェントロボットシステムの研究の動向と展望”. 日本ロボット学会誌 10.4, pp. 428–432, (1992).

[2] Muning Wen et al.: “Multi-agent reinforcement learning is a sequence modeling problem”. *Advances in Neural Information Processing Systems* 35, pp. 16509–16521, (2022).

[3] John Schulman et al.: “Trust region policy optimization”. *International conference on machine learning*, pp. 1889–1897, (2015).

[4] John Schulman et al.: “Proximal policy optimization algorithms”. *arXiv preprint arXiv:1707.06347*, (2017).

[5] Chao Yu et al.: “The surprising effectiveness of ppo in cooperative multi-agent games”. *Advances in Neural Information Processing Systems* 35, pp. 24611–24624, (2022).

[6] Jakub Grudzien Kuba et al.: “Trust region policy optimisation in multi-agent reinforcement learning”. *arXiv preprint arXiv:2109.11251*, (2021).

[7] Junyoung Chung et al.: “Empirical evaluation of gated recurrent neural networks on sequence modeling”. *arXiv preprint arXiv:1412.3555*, (2014).

[8] Alex Graves and Alex Graves: “Long short-term memory”. *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, (2012).

[9] Ashish Vaswani et al.: “Attention is all you need”. *Advances in neural information processing systems* 30, (2017).

[10] Alexey Dosovitskiy et al.: “An image is worth 16x16 words: Transformers for image recognition at scale”. *arXiv preprint arXiv:2010.11929*, (2020).

[11] Nicolas Carion et al.: “End-to-end object detection with transformers”. *European conference on computer vision*, pp. 213–229, (2020).

[12] Lili Chen et al.: “Decision transformer: Reinforcement learning via sequence modeling”. *Advances in neural information processing systems* 34, pp. 15084–15097, (2021).

[13] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly: “Pointer networks”. *Advances in neural information processing systems* 28, (2015).

[14] Wouter Kool, Herke Van Hoof, and Max Welling: “Attention, learn to solve routing problems!” *arXiv preprint arXiv:1803.08475*, (2018).

[15] Hao Lu, Xingwen Zhang, and Shuang Yang: “A learning-based iterative method for solving vehicle routing problems”. *International conference on learning representations*, (2019).

[16] Xinyun Chen and Yuandong Tian: “Learning to perform local rewriting for combinatorial optimization”. *Advances in Neural Information Processing Systems* 32, (2019).

[17] Michael L Littman: “Markov games as a framework for multi-agent reinforcement learning”. *Machine learning proceedings 1994*, pp. 157–163, (1994).

[18] John Schulman et al.: “High-dimensional continuous control using generalized advantage estimation”. *arXiv preprint arXiv:1506.02438*, (2015).

[19] Volodymyr Mnih et al.: “Human-level control through deep reinforcement learning”. *nature* 518.7540, pp. 529–533, (2015).

[20] Mikayel Samvelyan et al.: “The starcraft multi-agent challenge”. *arXiv preprint arXiv:1902.04043*, (2019).

[21] Benjamin Ellis et al.: “SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning”. *arXiv preprint arXiv:2212.07489*, (2022).

[22] Adam Paszke et al.: “Pytorch: An imperative style, high-performance deep learning library”. *Advances in neural information processing systems* 32, (2019).