マルチエージェント協調問題における２段階報酬配分法を用いた(深層)強化学習による協調的行動の発現と解析

宮下　裕貴
(技術研究所)

Coordinated Behavior for Sequential Task using Two-Stage Reward Assignment in Multi-agent Systems

Yuki Miyashita

本稿では、マルチエージェントの段階的協調タスク問題において、深層強化学習への報酬配分方法がエージェント間の協調行動発現に与える影響を調査した結果を報告する。エージェント間の協調は、マルチエージェント分野において主要な研究課題である。近年、課題の解決に向けてマルチエージェントシステムに深層強化学習を適用する技術が注目を集めるが、エージェントの行動に伴う報酬(評価)が協調行動、特に分業や組織的協調行動の発現に与えるメカニズムは明確化されていない。本研究は、抽象的な協調タスク問題を題材に、報酬分配の割合変化より異なる協調構造の発現を確認した。

Recently, multi-agent deep reinforcement learning (MADRL) has been studied to learn actions to achieve complicated tasks and generate their coordination structure. The reward assignment in MADRL is a crucial factor to guide and produce both their behaviors for their own tasks and coordinated behaviors. However, it has not been sufficiently clarified the reward assignment in MADRL's effect on learned coordinated behavior. To address this issue, using the sequential tasks coordinated delivery and execution problem, we analyze the effect of various ratios of the reward given for the task that agent is responsible for.

1. はじめに

マルチエージェントシステムにおいて、協調行動の実現は主要な研究課題の一つである。複数エージェントの協調制御システムの設計には、環境構造、各エージェントの能力、タスク構造などのシステム全体を包括した理解が必要となるが、システム設計時に全体を把握するのは困難である。加えて、相互作用のある複雑で動的な環境では、エージェントの協調行動を予め静的に定めたシステム設計は非効率となりうる。そのため、エージェントの自律的学習による協調行動の創発と動的な環境下での協調行動の柔軟な適応が望ましい。

エージェントの自律的学習による協調行動実現の手法に強化学習(Reinforcement Learning、RL)がある。マルチエージェントシステムにRLを適用するとき、環境の大きさやエージェント数増加に伴う状態テーブルの爆発的な増大による学習の非効率性や、相互にエージェントが学習する状況での他エージェントの行動の変化より生ずる不安定性と不確実性を考慮した学習メカニズムが必要となる。特に、エージェントの協調行動の学習には、自己に割り当てられたタスク実行とシステムの観点から効率性のバランスを考慮する必要があり、それらを得るための適切な報酬割り当てが極めて重要となる。単純で変化の小さい環境では、エージェントの協調行動に対する適切な報酬割り当ての設計は容易だが、動的で複雑な環境ではエージェントの協調行動による貢献を明確にし難く、協調行動を促す報酬割り当ての設計方法は明らかでない。

近年の研究では、単一エージェントに深層強化学習(Deep Reinforcement Learning, DRL)を適用し、ロボットの制御[1)]やビデオゲームの操作[2)]など様々な分野で成果を上げている。マルチエージェントシステムにおいても、動的で複雑な環境下でのエージェント間の協調行動を実現するためにDRLを適用したMulti-Agent Deep Reinforcement Learning (MADRL)の研究がいくつか提案されている[3)]。一般にDRLでは、適切な

行動価値を学習するために繊細な報酬設計が要求される。そのため、MADRLにおいて各エージェントが適切に行動価値を学習するには、自己のタスク処理に必要な行動と、他エージェントとの協調的な行動との均衡が取れた報酬割り当てが必要となる。しかし、これまでMADRLにおける報酬割り当ての協調的行動の発現に与える影響や協調行動がもたらすシステムの効率化の仕組みは不明瞭であった。

そこで本研究は、マルチエージェントシステムの自律分散制御において、異なるタイプのエージェント同士が協力してある順序に基づいてタスクを完遂する環境の下、遅延を導入した報酬配分方法と、その配分割合の変化に伴う協調行動の発生の有無とその特性を調査する。本実験環境は、建設現場におけるロボット施工システムへの適用を想定して、現場内で異なる役割を有する2種類のロボットが互いに協調し、順にそれぞれの役割に応じたサブタスクを限られた時間制約のもと完遂する作業を抽象化したものである。ここで、初めのロボット(エージェント)から見ると、サブタスクの真の完了は、その次のエージェントのサブタスクの完了後となり、自己のサブタスクの実行とタスクの完了には遅延が存在する。そこで、我々はサブタスク実行とタスク完了に伴うエージェントへの報酬の配分に着目し、自己のサブタスク実行のための効率的な行動学習と協調行動の学習を両立するエージェントへの報酬配分手法を導入する。加えて、この報酬配分手法の導入に伴い、学習データのサンプリングに用いるExperience replayも拡張する。本拡張に基づき学習から発現した協調行動の効率性、遅延して得られる報酬の割合が協調行動に与える影響を解析する。

本論文では、提案手法を本問題に適用することで効率的なサブタスク処理行動と協調的な振る舞いの学習を両立すること、また、報酬の割り当て方の違いからエージェントは異なる振る舞いを発現させることを述べる。例えば、2種類のエージェントが協力してタスクを完遂したときのみ報酬を割り当てたとき、エージェントは自己のサブタスク処理に必要な行動を十分に学習できなかった。さらに、サブタスク実行時に割り当てる報酬額に伴って、学習の収束に時間がかかるが協調性を重視した行動を発現する傾向や、自身のサブタスク処理の行動を効率的に学習する一方で協調的な振る舞いを軽視する傾向があった。これらより、MADRLの協調的タスクの実行問題において、学習の収束速度と協調的行動の発生は二律背反の関係にあると考えられる。また、学習の進行に合わせて報酬配分を変化させ、学習の効率性と協調行動の発現を備える手法を提案し評価する。

2. 関連研究

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を同定する強化学習の研究が多く報告されている[4),5)]が、中でも協調問題におけるエージェントへの報酬配分は極めて重要なため、それに関する多くの研究が存在する[6),7)]。例えば、文献[6)]では各エージェントが独立して行動を決定する環境において、エージェントの貢献に応じて報酬を分配する手法を提案した。文献[8)]は、エージェントの報酬をシステムへの貢献具合と個々の状態より算出される報酬を組み合わせた報酬分配手法を提案し、グリッド環境での探索問題において性能を向上させた。

近年、強化学習とDeep Learningを組み合わせたDeep Reinforcement Learning (DRL)を活用した研究が様々な分野で成果を上げている。例えば、文献[2)]は、環境が複雑となる3DのFPSゲームにDRLを適用し高い効果を示した。文献[1)]では、ロボットのアーム操作にDRLを適用し、雑多に配置された物体群から適切に物体をつかみ上げることに成功している。これら成功を受け、文献[9)]は学習効率を高めるニューラルネットワーク構成を提案した。

マルチエージェントシステムにDRLを適用したMulti-Agent Deep Reinforcement Learning (MADRL)の研究が報告されている。例えば、文献[10)]は蓄積した学習データのデータサンプリングを工夫することでマルチエージェントシステムの非定常性に対応した効率的な学習手法を提案した。また、MADRLにおける報酬割り当て手法の関連した研究も注目を集めている[3),11),12)]。文献[12)]では、様々な実験環境での検証を通して、個々に報酬を分配する場合と、全体で報酬を共有する場合と競争的な報酬割り当ての性能をそれぞれ評価している。文献[11),]はDRLのActor-Critic手法とDifference reward手法を組み合わせ、分散環境での協調的タスクにおける効率的な学習手法を提案した。しかし、これらMADRLの報酬割り当ての研究では、複数のエージェントが順に作業することで完了できるタスクに対してのエージェント間の報酬配分に着目しているが、本研究では各エージェントの行動とタスク完了時の2段階で同一エージェントへ報酬を配分することを提案している。さらに上記の研究では、報酬配分が

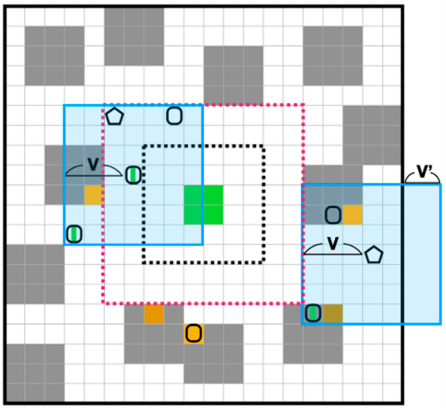
及ぼす協調行動の内容には言及しておらず、効率化の仕組みは不明である。一方、本研究では各エージェントへの2段階報酬の配分とそれによるエージェントの協調行動への影響を調査する。

3. 問題設定とモデル

3.1 環境とエージェントのモデル

本研究で想定するマルチエージェント協調問題は、建設現場において運搬ロボットが資材を運搬し、運ばれた資材を制限時間内に施工ロボットが処理(具合的には、運搬された資材を用いた資材の取り付けなどを想定)する環境を抽象化したものである。本問題において、タスクは運搬のサブタスクと施工のサブタスクの2つで構成され、制限時間内にサブタスクがこの順序に従って実行されたときタスク完遂とする。それぞれのサブタスクは、役割の異なる2種類のエージェントにより実施される。資材運搬の役割を有する運搬エージェントは、資材が供給されるエリアで資材を積み、施工箇所への運搬を目的とする。施工エージェントは、運搬された資材がある場所を探し、その資材を用いて施工する。記述簡略化のため、以下より運搬と施工を各サブタスクと定義し、施工されたとき、その場所のタスクの完了と定義する。運搬エージェントにより配置された資材は、存在可能な有効時間を有し、時間を過ぎると使用不可となり環境から除去されるものとした。実際の資材には有効時間が存在しないが、資材を長い時間放置することは効率が悪いため、今回はこのような設定を導入する．このような設定のため、運搬エージェントが配置した資材が除去される前に、施工エージェントはサブタスク(施工)を実行しなければならない。エージェントは環境内の全施工箇所のタスクが完了するまで上記の動作を継続する。

図－1 に本問題が想定するシステム環境例を示す。システム環境は N×N の格子とし、図中の黒の五角形が施工エージェント、黒の四角形が運搬エージェントを表す。資材運搬中の運搬エージェントは、四角内に緑の線を追記している。運搬エージェントによる資材拾い上げが可能な資材供給エリアを緑のセル、運搬エージェントの資材配置を可能とする場所(施工箇所)を灰色のセル(未施工セル)、資材が配置された未施工のセル(施工可能セル)を黄色で表す。資材の存在可能な時間を黄色の濃さで表し、残りの存在時間が短くなるほど灰色に近づける。



図－1 システム環境例

3.2 問題設定

本研究の協調問題を$\langle I,N,m,E,\{S_i\}_{i\in I},\{A_i\}_{i\in I}\rangle$と表す。ここで、2種類のエージェントを合わせたエージェント集合を $I=\{1, \cdots, n\}$、環境のグリッドサイズを N、環境内の未施工セルの数を m、セルとエージェントの状態を含めた環境の状態の集合を $E(\ni e)$と表す。時間は離散時間を導入し、単位をステップとする。時刻 t において、エージェント $i\in I$ は環境 $e_t\in E$ から自己を中心とした局所領域 $s_{i,t}$ を観測し、この状況の組を $S_i\ni s_{i,t}$ とする。このとき、エージェントは環境から正確に状況観測できるものとする。エージェントの行動の組を $A=A_1\times\cdots\times A_n\ni a_t=(a_{1,t},\ldots,a_{n,t})$とし、エージェントの種類に関わらず、エージェントは5つの行動 $A_i=A=\{up,right,down,left,work\}$から時刻ごとに行動を決定する。複数のエージェントが同時に同じセルへの移動を試みた場合、目的のセルへ移動するエージェントが無作為に選択され、残りのエージェントは元のセルに留まる。本環境では、エージェント同士の通信はなく、他エージェントの次行動を事前には知れない。エージェントの *work* 行動はエージェントの種類で異なり、詳細は次章で後述する。

エージェント i が時刻 t の環境 e_t において他エージェントと共に行動の組 $a_t\in A$ を実行したとき、環境は次状態 $e_{t+1}\in E$ へと推移し、i は報酬 $r_i(e_t,a_t)$を受け取る。エージェントに与える報酬の配分方法はエージェントの種類により異なり、詳細は 4.2 節に記載する。本研究では、各エージェントが有するDeep Q-Network(DQN)を用いて、サブタスク処理とタスク完遂から受け取る報酬の最大化を目的とした行動価値と方策 $\pi_i: S_i\rightarrow A_i$ を学習する。特に我々は、報酬配分の違いより生ずる施工エージェントのサブタスク処理(施工)を促す運搬エージェントの協調的行動の創発に着目する。

3.3 エージェントモデル

本環境では、始めに資材供給セルを環境の中央に配置する(**図－1** を参照)。次に、**図－1** 内の黒の点線で囲まれた6×6の範囲内にエージェントをランダムに配置し、3×3 のセルから構成される施工エリアを紫色の点線の枠外にランダムかつ重ならないように K 個配置する。配置が完了した後、エポックを開始し、全エージェントは自己の方策に基づいて行動を開始する。

時刻 t ごとに, 各エージェント $i \in I$ は環境 e_t で観測した状態 $s_{i,t}$ と方策 π_i に基づいて行動($a_{i,t}=\pi_i(s_{i,t}) \in A_i$)を実行する。移動行動を選択した場合、エージェントは隣接セルに移動するが、work 行動はエージェントの種類によって異なる。

1) 運搬エージェントの work 行動 : 運搬エージェントの役割は、資材供給エリアから資材を取得し、施工エージェントがサブタスク(施工)を実行できるように資材を適切に配置することである。エージェント i が資材供給エリアに移動したとき、i は自動的に資材 ψ_i を拾い上げ、未施工セルへの移動を開始する。ここで、i は資材を一つのみ運搬可能とし、供給エリアの資材は枯渇しないものとする。その後、ステップ t_d において i が未施工セル上で work 行動を実施(サブタスク実施)したとき、i は ψ_i をセルの上に配置し、このセルは施工可能セルに変わる。ここで配置された資材を $\psi_{i,td}$ と表す。配置された資材の存在可能時間を μ とし、時刻 $t_d+\mu$ までに資材 $\psi_{i,td}$ が施工エージェントによって利用されなければ、資材 $\psi_{i,td}$ は取り除かれ、その施工可能セルは未施工セルに戻る。サブタスクを実行した運搬エージェントへの報酬割り当てについては、4.2 節に詳述する。

2)施工エージェントの work 行動 : 施工エージェントの役割は、運搬された資材が使用不可となる前に、それを使ってサブタスク(施工)を実行することである。施工エージェント j は、移動しながら資材が配置されたセルを探し、そのセル上で資材 ψ_i を用いた *work* 行動を実施することでタスクが完遂される。タスクを完遂したセルは、完了セルに変化する(**図－1** の白色のセル)。時刻 t_e にタスクが完了すると、j は報酬 $r_{j,te}$ を受け取り、加えて j が使用した資材 $\psi_{i,td}$ を配置したエージェント i にも追加で報酬が割り当てるものとする。報酬割り当ての詳細は、4.2 節に詳述する。

上記に示したエージェントの行動は、規定の時刻 H まで繰り返し終えるか、全ての未施工セルが完了セルに移行するまで繰り返す。これを 1 エポックと呼ぶ。1 エポック終了後、環境は初期化され次のエポックを実施し、規定の施行エポック数 $F_e(>0)$ 回繰り返す。

4. 提案学習モデル

4.1 観測情報と Deep Q-Network

DQN は、エージェント i が観測した状態が s_i のとき、それをニューラルネットワークへの入力として近似された行動価値関数 $Q_i(s_i,a_i)$ を求める手法である。一般的な強化学習では、i の観測状態を s_i、そのときの行動を a_i としたとき、累積報酬が最大となるように行動価値関数 $Q_i(s_i,a_i)$ を学習する。ニューラルネットワークを使った行動価値関数の学習には、報酬からネットワークパラメータ群 θ を更新する必要がある。そこで、ある時刻 t の i の DQN のネットワークパラメータ群 $\theta_{i,t}$ を、以下式の平均二乗誤差法による損失関数 $L_{i,t}(\theta_{i,t})$ の勾配からこの値を小さくするよう更新する。

$$L_{i,t}(\theta_{i,t}) = \mathbb{E}_{(s_i,a_i,r_i,s_i')}[(r_i + \gamma \max_{a_i'} Q_i(s_i', a_i'; \theta_{i,t}); \theta_{i,t}^-) - Q_i(s_i, a_i; \theta_{i,t}))^2],$$

ここで、r_i は a_i に伴う状態推移による報酬を表し、$\gamma \in [0,1]$ は割引率である。この式では、メインネットワークのパラメータ群 $\theta_{i,t}$ とターゲットネットワークのパラメータ群 $\theta^-_{i,t}$ を利用して損失関数を算出する double DQN[9]手法を使って $\theta_{i,t}$ を更新する。ここで、$\theta^-_{i,t}$ は $\theta_{i,t}$ へ更新する前のパラメータである。上記式において、メインネットワーク $\theta_{i,t}$ で行動を決定し、ターゲットネットワーク $\theta^-_{i,t}$ に基づく行動価値関数を用いて $\theta_{i,t}$ を時刻 η ごとに更新する。この $\theta^-_{i,t}$ は 1 エポックごとに $\theta_{i,t}$ からコピーされる。

一般的に、エージェントは観測状態に基づいて方策を決定するが、本研究では時刻 t のエージェント i の観測情報 $s_{i,t}$ と i が内部に持つ情報を合わせた $v_{i,t}$ を *view* と呼び、i の Q_i と方策 π_i を以下の式で拡張する。

$$Q_i : \mathcal{V}_i \times A_i \longrightarrow \mathbb{R}, \text{ and } \pi_i : \mathcal{V}_i \longrightarrow A_i,$$

ここで、V_i は i の *view* の集合を表す。$v_{i,t}$ に関しては、4.4 節にて詳述する。

4.2 2 段階報酬配分手法

強化学習によるエージェントの行動学習は、受け取る報酬の最大化を目的として学習するため、エージェントへの報酬割り当ては適切な行動を誘発する重要なファクターとなる。特に我々の問題では、サブタスクの逐次実行を必要とするため、最初にサブ

タスクを実行する運搬エージェントと次のサブタスクを実行する施工エージェントが、自己のサブタスク処理の行動学習のみならず、タスク完了に向けた協調行動を学習する必要がある。従って、これら行動を適切に学習するための個々エージェントへの報酬割り当てが重要となる。そこで我々は、それらの学習を両立させるためにサブタスク実行とタスク完了に報酬を配分する2段階報酬配分手法を提案し、それら報酬の配分変化によるエージェントのサブタスク処理行動の学習速度と、協調行動の発生と特徴を調べる。

2段階報酬配分手法では、最初にサブタスクを実行するエージェントは、自己サブタスクの実行と後続するサブタスクの実行有無に基づき2段階に分けて報酬を受け取る。まず、エージェント i は時刻 t_d でサブタスクの完了時に報酬 $r_1(t_d)(\geq 0)$を受け取る。その後、後続するサブタスクが時刻 $t_e = t_d + a$ でタスクが完了したとき、i は新たに報酬 $r_2(t_e)$を受け取る。つまり、i が実行したサブタスクと紐づくタスクが一定期間内の t_e で完遂したとき、i は最終的に報酬 $r_1(t_d)+r_2(t_e)$を受け取り、t_d に取った行動を強化する。

一方で、i が実行したサブタスクと紐づくタスクが完遂しなかったとき、i は報酬 $r_1(t_d)$のみを受け取る。

また、$r_1(t_d)$と $r_2(t_e)$の比を変えることも可能である。特に、これらの和を一定として $r_1(t_d)$を学習の進行に合わせて減少させる本手法を gradually decayed reward(GDR)と呼ぶ。具体的には、F_r エポックごとに r_1 を減少させ、h エポック目の $r_1(h)$を、

$$r_1(h) = r_a - \delta_r \cdot \lfloor h/F_r \rfloor$$

と定義する(右辺が負のときは $r_1(h) = 0$ とする)。GDR の導入により、運搬エージェントは学習初期に自己のサブタスク処理の振る舞いを学習し、学習が進んだ後にはタスク完了につながる協調行動の学習を促すことで、自己タスク処理とタスク完了のための協調行動を効率的に学習することを期待する。

4.3 Experience Replay の拡張

我々は、2段階報酬配分手法を DQN の学習に適応するため、Experience replay を拡張する。

Experience replay は、学習に利用する経験の連続の相関関係より生じる過学習を排除するため、経験をランダムにサンプリングする手法である。マルチエージェントシステムにおいて、各エージェントは独立して行動を学習するため、非定常な環境下で学習をしなければならず、経験間の相関を排除する Experience replay は有効と考える。提案する2段階報酬配分手法では、受け取る報酬が将来の他エージェントの行動により変わるため，これに対応できるよう Experience replay を拡張する。

エージェント i の時刻 t の replay memory を $D_{i,t}$ と表す。報酬が確定していない運搬エージェント i の経験が学習に使われることを防ぐため、i は時刻 t での経験 $c_{i,t} = (s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1})$を $D_{i,t}$ ではなく一時待避キュー $P_{i,t}$ に保存する。ここで、$P_{i,t}$ の最大サイズは $M_p(\geq \mu > 0)$とし、一時待避キューが $|P_{i,t}| > M_p$ を満たすとき、キューの先頭の経験データは $D_{i,t}$ に移動する。i の行動ごとの報酬の大半は0であるが、i が時刻 t_d での自己のサブタスク実行により資材 ψ_{i,t_d} を配置したとき, 報酬 $r_{i,t_d} = r_1(t_d) \geq 0$ を受け取り、対応する c_{i,t_d} を P_{i,t_d} に追加する。その後、時刻 t_e に他エージェントが ψ_{i,t_d} を用いてサブタスクを実行し、そのタスクが完遂したとき、i は報酬 r_2 を受け取り、P_{i,t_e} の中にある c_{i,t_d} の報酬を $r_{i,t_d} = r_1(t_d)+r_2(t_e)$ に変更する。配置された資材 ψ_{i,t_d} は存在可能時間 μ を有するため、t_e は $t_d+\mu \geq t_e > t_d$ を満たす。配置された資材 ψ_{i,t_d} が $t_d+\mu$ までに利用されなかった場合、経験データ c_{i,t_d} の報酬 $r_{i,t_d}(= r_1(t_d))$の変更はなく、そのまま c_{i,t_d} を D_{i,t_d+M_p+1} に移動させる。施工エージェントには追加の報酬がないため、一時待避キューを利用せず、個々が有する replay memory に自己の経験を追加する。

この結果、ステップ t での運搬エージェント i の replay memory は $D_{i,t} = \{c_{i,t-M_p-M_d}, \cdots, c_{i,t-M_p-1}\}$ となる。ここで $M_d(>0)$は、replay memory の保存容量を表す。このデータセット $D_{i,t}$ に保存された経験から、ミニバッチ数 U のランダムサンプリングよりミニバッチ $U(D_{i,t})$を生成し、以下式の $L_{i,t}(\theta_{i,t})$を最小化するようにニューラルネットワークのパラメータ $\theta_{i,t}$ を更新する。また、パラメータの更新は η ステップごとに実施する。

$$L_{i,t}(\theta_{i,t}) = \mathbb{E}_{(s_i, a_i, r_i, s_i') \sim U(D_{i,t})} [(r_i + \gamma \max_{a_i'} Q_i(s_i', a_i'; \theta_{i,t}); \theta_{i,t}^-) - Q_i(s_i, a_i; \theta_{i,t}))^2]$$

その後、損失関数 $L_{i,t}(\theta_{i,t})$を最小化する。例えば、損失関数の勾配 $\nabla L_{i,t}(\theta_{i,t})$を求め、その勾配を使ったパラメータ $\theta_{i,t}$ の更新に RMSprop[13]を使用する。

4.4 エージェントの状態生成

エージェント i は、自己を中心とした $V_i(\geq 0)$の範囲の観測により、時刻 t での状態 $s_{i,t} \in S_i$ を得る。この状態 $s_{i,t}$ には、観測範囲内のセル、資材、他エージェント(資材保有の有無を含め)が含まれる。

状態 $s_{i,t}$は$(2V_i+1)\times(2V_i+1)$の行列を複数組み合わせた構成となる。例として、エージェントの観測範囲 $V_i=V$を青色の四角で**図－1**に示す。

エージェントの種類によらず、iは観測した状態 $s_{i,t}$と iの内部情報から view $v_{i,t}$を形成し、個々が有する DQN への入力情報とする。入力情報 $v_{i,t}$は、5 種類 7 つの行列で構成される。前提として、エージェントは環境の形状と大きさと、(例えば GPS などを用いて)環境内の自己位置を内部情報として保有し、この内部情報を用いて環境内の自己位置を表現する行列 $N\times N$を生成する。残り 4 種類の行列は全て観測状態 $s_{i,t}$から構成され、エージェント種類に関わらず一律の規則に従い 6×M×M の行列として生成される。

1 種類目の行列は、観測範囲内にある未施工セル(灰色のセル)の位置を 1、それ以外を 0 として行列に記述する。以下説明する行列においても、特徴を表すオブジェクトが存在しない場所は行列内に 0 として記述する。2 種類目の行列では、環境に配置済み資材を表す。資材には存在時間があるため、行列上には資材 $\psi_{i,t_{dno}}$の位置にその存在時間 $z_t=max((t-t_d)/\mu,0)$を記述する。3 種類目の行列は他エージェントの位置を表す。このとき，エージェントは他エージェントの種類を特定できないが、各エージェントに一意に割り振られた ID を複数の行列を用いて表現する。ここではエージェントの ID を 3 桁の数字(b_1,b_2,b_3)(数字は $b_k\in\{0,1,-1\}$)で表し、そのエージェントの位置に ID の数値を、3 つの行列に分けて b_iを記述し、エージェントが存在しない箇所は 0 と記述する。ここで、エージェントの存在しない箇所との区別ができない(0,0,0)の ID は割り当てから除外する。最後に、4 種類目の行列は運搬エージェントの資材保有の有無と資材供給セルを表現し、資材の保有するエージェントの位置と資材供給セルを 1 と記述する。運搬エージェントの自己の資材保有は、この行列の中心に 1 を記述し表現する。

エージェント iは、自己の過去の軌跡を内部情報として保存し、環境内の自己位置を表す行列に過去の軌跡を追加する。iの現在の自己位置を行列内に 1 と記述し、kステップ前の自己位置は $1\times\beta^k$ $(=\beta^k)$と記述する。ここで、$\beta\in[0,1)$は過去位置の軌跡の減衰率を表し、もし β^kが過去の軌跡の反映を決定づける閾値 δ_tを下回る場合は行列に記述しない。つまり、閾値 δ_tを下回る古いステップの iの位置は、行列内に反映されない。もしエージェントが過去の軌跡が残るセルに移動した場合は、そのセルにおける大きな値のみ行列に記述する。

表－1 ネットワーク構成

Layer	Input	Filter size	Stride	Activation	Next Layer
Conv-1.1	$M\times M\times 6$	2×2	1		Conv-1.2
Conv-1.2	$M\times M\times 32$	2×2	1		Max pooling-1
Max pooling-1	$M\times M\times 32$	2×2	2		FCN-1
Conv-2	$N\times N\times 1$	2×2	1		Max pooling-2
Max pooling-2	$N\times N\times 16$	2×2	2		FCN-1
FCN-1	$M/2\times M/2\times 32+$ $N/2\times N/2\times 16$			ReLu	FCN-2
FCN-2	512			ReLu	FCN-3
FCN-3	256			Linear	5

表－2 学習パラメータ

Parameter	Value
DQNの割引率 γ_q	0.95
の初期値$_i$ $=_{i,0}$	0.99999
の減衰率 γ	0.999999
の下限値$_l$	0.002
θ_iの学習タイミング η	8
RMSpropの学習率	0.00001
RMSpropのモーメンタム	0.90
RMSpropの $_{rms}$	1e-07
Experience Memoryの容量 M_d	2000
ミニバッチサイズ $\lvert U(D_{i,t})\rvert$	32

表－3 実験パラメータ

Parameter	Value
システムグリッド N	20
エージェント数 n_d+n_e	12
エージェントの観測範囲 V	3
未施工エリアの数 K	12
資材の存在可能時間 μ	6
運搬エージェントの報酬 r_1+r_2	1
施工エージェントの報酬 r_e	1
1エポック内のステップ数 H	600
エポック総数 F_e	13,000
軌跡の減衰率 β	0.9
軌跡の下限閾値 δ_t	0.05

4.5 ネットワーク構成

各エージェントは、個々に有する DQN を用いて自律的に行動を決定する。本実験で用いる DQN のネットワーク構成を**表－1**に示す。**表－1**に示すとおり、ネットワークは複数の convolutional layers(Conv)、max pooling layers と三層の fully connected network layers(FCN layers)で構成される。ネットワークには、エージェントの view で表す $6\times M\times M$のデータ(**表－1** の Conv-1.1)と $1\times N\times N$のデータ(表-1 の Conv-1.2)をそれぞれ入力する。こ

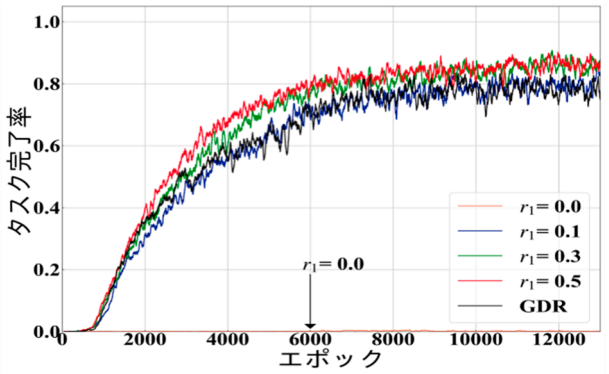
こで、エージェントの観測範囲 Vより $M=(2V+1)$ とする。分離している Conv は、FCN layer で結合され、FCN layer の出力はエージェントのアクションと紐付き、出力値を入力に対応した行動価値と考え、この値が最大となる行動を選択する。エージェントの行動決定戦略には、確率 $\varepsilon_{i,t}$でランダムに行動し、確率 $1-\varepsilon_{i,t}$で DQN の出力によって行動を決定する ε-greedy 戦略を用いる。学習初期はランダムに行動して様々な経験を集め、学習が進んだ後には行動価値による行動選択によって報酬を受け取る経験を獲得するために i ごとに ε_iを定め、その $\varepsilon_{i,t}$を指数的に減少させる。具体的には、$\varepsilon_{i,t}$をステップごとに $\varepsilon_{i,t}=\max\{\varepsilon_{i,t-1}*\gamma_\varepsilon,\varepsilon_l\}$と更新する。ここで、$\varepsilon_l\geq 0$ を ε_iの下限値とし、$\gamma_\varepsilon \in [0,1)$を ε_iの減衰率とする。

5. 実験

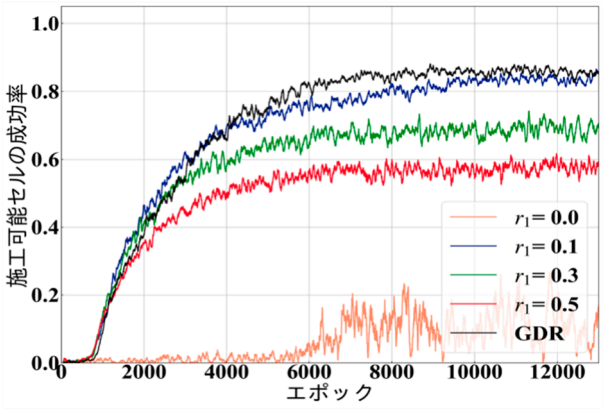
5.1 実験環境

本実験は、2 種類のエージェントが協調してタスクを完遂する問題において、4.2 節で述べた報酬配分手法を導入したときの性能評価と協調行動を解析する。始めに報酬 $r_1(t)=r_1$の違いの影響を測るため、r_1に 0 から 0.5 の値を設定し、未施工セルの施工完了割合を表すタスク完了率と、配置された資材の利用率を表す施工可能セルの成功率(資材が使用されたら成功、使用されずに資材が除去された場合は失敗)を用いる。ここで、タスク完了率から本実験におけるタスク処理の性能を測り、施工可能セルの成功率から種類の異なるエージェント間での協調行動の発現の割合を測っている。なお、$r_1+r_2=1$ に注意されたい。その後、各エージェントの行動の詳細分析から r_1 ごとの振る舞いの違いを確認する。

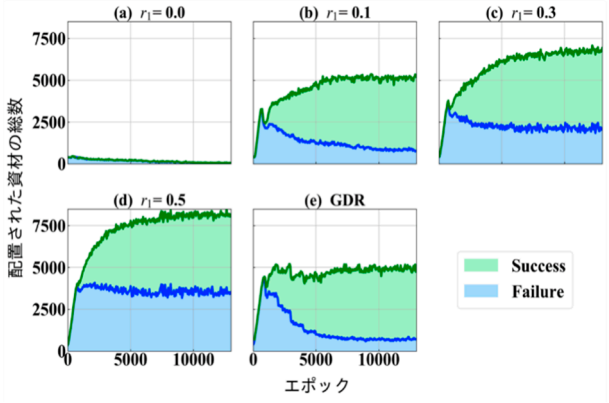
さらに、実験の進行に合わせて r_1 を減少させる GDR 手法を導入し、r_1 を固定した場合との性能やエージェントの振る舞いを比較する。なお、r_a は r_1 の初期値、δ_r は減衰率を表す。本実験では GDR のパラメータ値を $r_a=0.5$、$\delta_r=0.1$、$F_r=1000$ と設定し、5000 エポック以降は $r_1=0$ とする。実験に用いる他のパラメータを**表－2** と**表－3** に示す。本実験における未施工セルの数は 108(= 9×K(= 12))、運搬エージェント数は 8 体、施工エージェントは 4 体とする。これのエージェント数の差は、運搬エージェントによる資材供給エリアと施工セル間の往復にかかるコストから、施工エージェントとのコスト比を考慮した値である。



図－2 タスク完了率



図－3 施工可能セルの成功率



図－4 配置された資材総数の推移

5.2 性能比較

報酬 r_1 の違いによるタスク処理性能への影響を調査するため、タスク完了率から報酬 $r_1=0$、 0.1、0.3、0.5 と GDR の性能を比較する。**図－2** に 50 エポック区間ごとのタスク完了率の移動平均の推移を示す。**図－2** より、$r_1=0$ を除いた報酬配分においてタスク完了率の向上が見られる。$r_1=0$ では、エージェントは適切な振る舞いを学習できず、ほとんどの

タスクが完了できなかった。また、r_1=0.5 が最も早く収束したが、収束後のタスク完了率は報酬 r_1=0.3 と同程度である。GDR のタスク完了率は r_1=0.3 と r_1=0.1 の間に収束している。

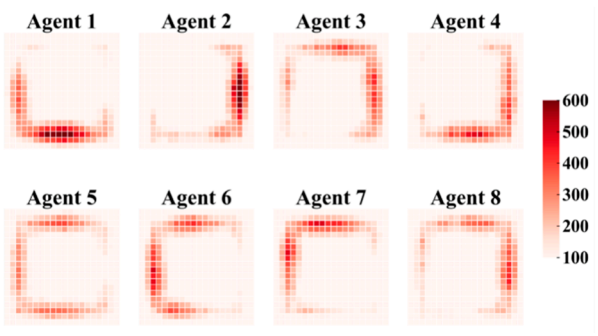
運搬エージェントのサブタスクが施工エージェントのサブタスク実行につながった割合、つまり施工可能セルの成功率を、50 エポック区間ごとの移動平均の推移として**図－3** に示す。成功率が高いほど、運搬エージェントの実行したサブタスクを無駄にしない協調行動が発現したことを示す。**図－3** より、報酬 r_1 を固定した実験の中で r_1=0.1 の成功率が最も高く、r_1 の値が大きくになるに従って成功率が減少する。GDR の成功率は r_1=0.1 より収束が早く、軽微な差であるが r_1=0.1 を上回る最も高い成功率となった。以上から GDR や r_1=0.1 のとき、運搬エージェントと施工エージェントは、資材の無駄が少ない協調的行動をとることがわかる。エージェントの振る舞いは次節にて分析する。

r_1=0.1 の成功率が低いのは、運搬エージェントが自己のサブタスク処理の行動を学習できず、施工エージェントもサブタスク実行の機会が少ないため、十分に学習できなかったためと考えられる。

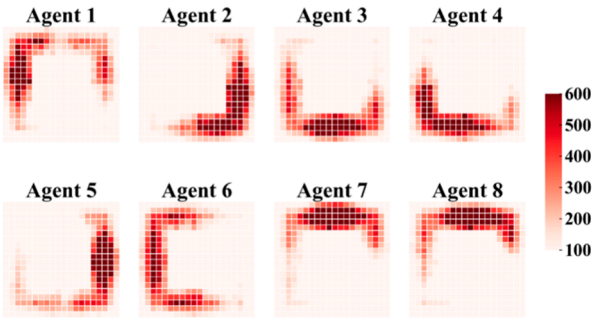
5.3　資材配置行動の分析

図－2 より、報酬 r_1 の違いに伴って特に協調的行動に差が生じることがわかった。そこで、エージェントの行動の分析から協調行動に差が生じた要因を論ずる。まず、運搬エージェントにより運搬された資材の施工エージェントによる利用有無をより詳しく分析するため、それぞれの報酬ごとの積層折れ線グラフを**図－4** に示す。50 エポックごとに運搬エージェントにより配置された資材を加算し、施工エージェントに利用された資材を緑、利用されなかった資材を青で表す。**図－4** より、r_1=0.5 のとき最も多くの資材が運搬エージェントにより配置されたが、配置された多くの資材が施工エージェントに利用されなかった。加えて、r_1 の値が小さくなるに従って配置される資材の総数が減少し、使用されなかった資材の数も大きく減少する。これは特に r_1=0.5 の場合(**図－4** (d))は、運搬エージェントは施工エージェントとの協調的振る舞いに関係なく自己のサブタスク処理から十分な報酬を得られたためと考えられる。

逆に r_1 が小さいと、施工エージェントのサブタスク実行に伴い、運搬エージェントは多くの報酬を獲得できる。**図－4** (b)より、配置された資材総数は少ないものの未利用の資材は少ないことから、運搬エージェントは自己が配置した資材の利用を促すため、



図－5 ノードごとのサブタスク処理回数(r_1=0.1)



図－6 ノードごとのサブタスク処理回数(r_1=0.5)

施工エージェントと何らかの協調的な振る舞いをしたと考えられる。GDR を用いた場合は(**図－4** (e))、学習の収束が早いだけではなく r_1=0.1 (**図－4** (b))より未利用の資材の数も少なく、最も資材を効果的に利用する行動を学習した。

5.3　資材配置行動の分析

次に、同じ種類のエージェント間での協調的な行動を調べるため、運搬エージェントが環境内のサブタスク(資材の配置)を実施した場所とそのときの環境状況を調べた。資材が設置された場所を 10,000 から 13,000 エポック間で調査し、エージェントごとの総設置数をヒートマップとして**図－5**(r_1=0.1)、**図－6**(r_1=0.5)に示す。ここで、濃い赤色のセルはそこでのサブタスク処理回数が多く、白色に近づくにつれてそのノードでのサブタスク処理回数が減少することを示す。**図－5**、**図－6** どちらからも、運搬エージェントは他エージェントとのサブタスク実施領域の重複を減らすように自律的に担当領域を定めることで、運搬エージェント同士でのサブタスク実施の競合を減少させる協調的な組織的分業を形成したことがわかる。ここで、**図－5** と**図－6** に赤色セル数の差があるが、**図－4** で示したように r_1=0.1 と r_1=0.5 では、運搬エージェントのサブタスク実行回数に 1.5 倍近い差があるためである。ページの都合上、r_1=0.3 と GDR のヒートマップは載せていない

が、r_1=0.1,0.5 と同様の協調的な組織的分業を形成し、併せて、施工エージェントにおいても、施工エージェント同士でのサブタスク実行の競合を減少させる同様な分業が形成されることを確認した。

5. 考察

今回の実験から、協調してタスクを遂行する問題においてエージェントの行動に伴い配分される報酬値の違いにより、エージェントは異なる振る舞いを形成した。本問題において、タスク完遂のために異なる種類のエージェント間の協調的行動と、自己のサブタスク処理に必要な行動の自律的学習が必要となる。2段階報酬配分手法の運搬エージェントの最初の報酬 r_1 が小さく、タスク完遂時の報酬が大きいとき、運搬エージェントは、自己のサブタスク実行を活かすように施工エージェントのサブタスク実行を促す協調行動を学習したが、学習の収束が遅かった。ただし報酬 r_1=0 のとき、運搬エージェントは自己のサブタスク処理の行動からは報酬を受け取れず、行動価値を学習する機会が乏しく、他のエージェントも含め行動を十分に学習できなかった。

一方、r_1 が大きいとき、運搬エージェントは自己のサブタスク実行で十分な報酬を得られるため、自己のサブタスク処理の学習の収束は早かったが、施工エージェントとの協調行動を十分には学習せず、運搬エージェントの行動の多くが無駄になった。

GDR を用いた報酬割り当てでは、運搬エージェントは学習初期に自己のサブタスク処理の振る舞いを学習し、その後に協調行動を学習した。運搬エージェントは学習初期の段階で多くの資材を配置するため、施工エージェントも学習の機会が多くなり、その後に、エージェント間の協調行動を効率よく学習できた。このため GDR を用いたときのサブタスクに紐づくタスクの完了率は最も高く、収束も早くなった。

同じ種類のエージェント同士の協調行動を分析すると、エージェントは自己のサブタスク処理の担当領域を他エージェントと重ならないように調整することで、エージェント間の競合発生を抑制する行動を生成した。このときの同種類のエージェント同士の協調的振る舞いは、報酬割り当ての値に関わらず発生している。

ページの都合上、実験結果は割愛するが、報酬 r_1 が正でもその値の違いより運搬エージェントの資材配置の行動が大きく異なることも確認した。報酬 r_1 が大きいとき、運搬エージェントは協調相手となる施工エージェントの観測の有無に関わらず資材を配置(サブタスク実行)する傾向があった。一方、GDR もしくは報酬 r_1 が小さいとき、運搬エージェントは観測範囲内に施工エージェントが入るまで待って資材を配置する傾向があった。これには協調行動と自己のサブタスクの完了のみを追求する行動のトレードオフを暗示している。

5. おわりに

本研究では、施工現場におけるロボット施工システム適用を想定した問題を用いて、異なるタイプのエージェント同士が協力してタスクを完遂する環境の下、自己のタスクの効率化と協調行動をともに生み出す2段階報酬配分法を提案し、その報酬配分が協調行動へ及ぼす影響を調べた。実験より、自己のサブタスク処理の報酬が多いと、自己のサブタスク処理の行動を効率的に学習するが、タスク完了に必要な協調行動を軽視する傾向があった。一方、自己のサブタスク処理の報酬が少ないとき、協調性を重視した行動を学習したが学習の収束には時間を要し、自分のサブタスク実行に慎重になるためシステムにとって非効率となった。これらの結果から報酬配分法 GDR を提案し、これを適用すると自己のサブタスク処理に必要な行動の効率的学習と協調行動を両立でき、システム性能の向上に寄与した。

今後は、GDR の r_1 の減少率とエージェントの振る舞いの関係性をより深く分析し、その後、より複雑なタスク構成、例えば多段階にサブタスクの逐次実行を有する問題における報酬配分の協調的動作への影響や GDR 手法の効用を調査したい。また、タスク処理に必要なエージェント数を増やした実験の実施に取り組む予定である。

謝辞

本研究を進めるに当たり、早稲田大学の菅原教授からは多大な助言を賜りました。厚く感謝を申し上げます。

＜参考文献＞

1) Gu, S., Holly, E., Lillicrap, T., and Levine, S.: Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Offpolicy Updates, Proceedings of International Conference on Robotics and Automation, pp.3389-3396, 2017
2) Lample, G. and Chaplot, D. S.: Playing FPS Games with Deep Reinforcement Learning, Proceedings of the AAAI Conference on Artificial Intelligence, pp.2140-2146, 2017

3) Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S.: Counterfactual Multi-agent Policy Gradients, Proceedings of the AAAI Conference on Artificial Intelligence, 2018
4) Shoham, Y., Powers, R., and Grenager, T.: Multi-agent Reinforcement Learning: a Critical Survey, Technical report, Stanford University 2003
5) Busoniu, L., Babuska, R., and De Schutter, B.: A Comprehensive Survey of Multiagent Reinforcement Learning, IEEE Trans. Systems, Man, and Cybernetics, Part C, Vol. 38, No. 2, pp.156-172, 2008
6) Tumer, K. and Agogino, A.: Multiagent Learning for Black Box System Reward Functions, Advances in Complex Systems, Vol. 12, No. 04n05, pp.475-492, 2009
7) Becker, R., Zilberstein, S., Lesser, V., and Goldman, C.: Solving Transition Independent Decentralized Markov Decision Processes, Journal of Artificial Intelligence Research, Vol. 22, pp.423-455, 2004
8) Devlin, S., Yliniemi, L., Kudenko, D., and Tumer, K.: Potential-based Difference Rewards for Multiagent Reinforcement Learning, Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems, pp.165-172, 2014
9) Van Hasselt, H., Guez, A., and Silver, D.: Deep Reinforcement Learning with Double Q-Learning, Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 2, p.2094-2100, 2016
10) Palmer, G., Tuyls, K., Bloembergen, D., and Savani, R.: Lenient Multi-agent Deep Reinforcement Learning, Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, pp.443-451, 2018
11) Nguyen, D. T., Kumar, A., and Lau, H. C.: Credit Assignment for Collective Multiagent RL with Global Rewards, Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp.8113-8124, 2018
12) Jiang, J. and Lu, Z.: Learning Attentional Communication for Multi-agent Cooperation, Proceedings of Advances in Neural Information Processing Systems, pp.7254-7264, 2018
13) Tieleman, T. and Hinton, G.: Divide the Gradient by a Running Average of its Recent Magnitude, *COURSERA: Neural networks for machine learning*, Vol. 4, No.2, pp.26-31, 2012