Vol.2022-ICS-205 No.10
2022/2/21

時間制限付き半順序作業における協調行動学習のための漸進的報酬設計の提案

小國 祥寛[1,a)]　宮下 裕貴[1,b)]　菅原 俊治[1,c)]

概要：本稿では，異なる能力と役割を持つ複数のエージェントが，決められた順序で作業することで完了するタスクにおいて，自分の作業の効率化の学習と自分の前後に作業するエージェントとの連携行動の学習を両立させる手法を提案する．近年，深層強化学習の研究が活発に行われているが，エージェント間の協調行動の学習については未だ十分とは言えない．特にエージェントが独立した深層ネットワークを持つ分散的なマルチエージェント強化学習 (MADRL) では，個々の作業に与える報酬の体系が協調行動に与える影響については不明な部分も多い．本稿で対象とする順序付きの作業では，特に初めのエージェントからは，かなり遅れてタスクが完了し，すでに他のタスクの作業中となることも多い．そのため，たとえば，自分の担当部分の完了時に報酬を与えるとタスクの完了という本来の目的と異なった報酬体系となり，自己中心の作業に特化し，他の行動を考慮しなくなる．一方で，全体の作業が終わったときにのみ報酬を与えると，自分の作業の学習が進まないばかりではなく，その後のエージェントの学習機会も失う．このような課題に対し，本提案では，報酬を 2 分割し，それを学習の遂行とともに変化させ，自分の作業の効率と協調行動をともに実現し，最終的には本来の目的と合致するように関連するすべての作業が完了したときにのみ報酬を与えるものとする．実験により，報酬比と収集した行動を比較し，提案手法が自己の作業と協調・連携行動のバランスのとれた行動学習を実現することを示す．

1. はじめに

インターネットや IoT が普及する中で，単一のコンピュータやプログラムだけで構成されるシステムは限られており，複数のプログラムの連携によりサービスが達成されることが多い．そのようなシステムを集中的に制御するのは効率の面で不利となることもあり，独立した自律ノード（エージェント）による自律的分散システムが望まれる．しかしエージェント間の連携・協調行動はシステムごと場面ごとに依存する多種多様なものであり，事前に全てを把握してシステムを構築することは難しく，状況に合わせて適切に行動する学習法が求められる．しかし，エージェントが認知する環境には他のエージェントも含まれ，その状態数は極めて大きく，さらに他のエージェントの学習の進行に依存して最適な行動は変化し，学習の障害となることもある．

一方，強化学習の行動価値関数や方策の学習にニューラルネットワークを使用する深層強化学習はビデオゲームのような状態数や行動のパターンが多い問題でも適切な行動を学習できることが知られている [1], [2]．そこで，深層強化学習の機能を持つ複数のエージェントを同一の環境内で動作させ，全体の効率を上げるマルチエージェント深層強化学習 (multi-agent deep reinforcement learning，MADRL) が研究され一定の成果をあげている．しかし，システム全体としての効率にはエージェントの自分自身の報酬のための行動だけでなく，関連する他のエージェントとの協調・調整行動も大きく影響し，それら全体をバランスよく獲得する学習法についての研究は未だ不十分な状況である．

本研究では，複数台のロボット（エージェント）による時間制限のある逐次的な作業（タスク）を考える．たとえば，建築現場の施工では，複数の資材（部品）の運搬とそれらの配置や固定などを担う異なるロボットが，ある順序（半順序）でそれぞれの作業（部分タスク）を一定時間内に遂行する．このようなエージェントによる複数段階に分解できる共同作業は，様々な場面で想定しうる．しかし，初期の部分タスクを担当するエージェントの観点からは，タスクの完了はそれ以降のエージェント達の成否に依存し，また得られる報酬もかなり遅れて到着し，到着時には別のタスクに取り組んでいることも多い．他方，後半のエージェントは，それ以前のエージェントの行動の学習が進まないと，部分タスクを実行する機会すら得られない．そのため

1 早稲田大学
169-8555 東京都新宿区大久保 3-4-1
a) y.oguni@isl.cs.waseda.ac.jp
b) y.miyashita@isl.cs.waseda.ac.jp
c) sugawara@waseda.jp

ⓒ 2022 Information Processing Society of Japan

タスクの成否に貢献する行為は自分の作業（部分タスク）の効率のみならず，続くエージェントの効率性やそれらとの協調行動にも依存し，特に作業の前後のエージェントとの協力行動も同時に強化する必要がある．

しかし，一般的に用いられる固定的で完了時のみの報酬 (global reward) の枠組みでは，MADRL により各エージェントのシステム全体への貢献や他のエージェントへの影響を考慮し，自己の部分タスクの効率性と他エージェントとの協調行動をバランスよく学習することは容易ではない．特に，本研究で対象とする複数の部分タスクをある順序で実施するタスクでは，初期に作業したエージェントの行動は同じでも，後のエージェントの行動如何で無駄に終わることもあり，その成否も遅延して判明する．そのため，自己の行動が真に報酬獲得に貢献したかを判断しにくい．また，他のエージェントの行動は学習と共に変化し，それによって自己の適切な行動も変わる [3]．

MADRL における報酬を工夫して協調行動を学習する手法はいくつか存在する．たとえば，貢献度配分問題 (credit assignment problem) では，協調作業において各エージェントの貢献度に応じて合理的な手法で報酬を配分する．しかし本論文が想定する順序付きのタスクは，どの部分タスクが欠如しても達成はできず報酬はゼロとなる．また，エージェントの報酬割り当てを調整して協調/ 敵対の問題に対処する研究 [4], [5] がある．しかし，本研究が対象とする順序付きのタスクでは，自己の部分タスクの効率化と，それと前後するエージェントとの協調行動の両立が必要である．たとえば，自己の作業効率や報酬を少々犠牲にしても協調行動が全体の成功率を向上できるなら，必要な行動としてバランスをとるべきであり，そのような行動を学習すべきである．

このような課題に対し，時間制限のある 2 つの逐次的な部分タスクからなるタスクを対象とした，効率的で無駄のない行動を学習させる研究がある [6]．この研究では，報酬を 2 分割し，その配分を動的に変化させることで，最終的には global reward のみとなるが，バランスよく自己の部分タスクの効率と協調行動を学習できることを示した．しかし，2 つの部分タスクという単純な構成を対象としており，より複雑なタスク構造を持つもの，たとえばより多くの部分タスクを逐次的に行うタスクあるいは半順序関係のあるタスクは想定されておらず，具体的な報酬分割法は議論されていなかった．

そこで本研究では，半順序関係に従う多段のタスク構造をモデル化するとともに，上記の提案手法を拡張し，より複雑なタスク構造を持つ場合にも適用できる手法を提案する．その後，実験によって，自己の部分タスクの効率化と協調行動の学習を両立できることを示す．また，報酬が global reward のみの場合や，報酬を 2 分割するがそれらの値が固定されている場合と比較し，学習行動の差を分析して，効率化を達成した行動の内容について議論する．

2. 関連研究

強化学習ではエージェントが得る報酬関数を予め決める必要があり，その設計が学習の成否に影響する．しかし，学習対象ごとに適切な報酬関数を決めることは困難であり，それを解決する多くの研究が存在する．たとえば，[7] では，学習を促進するために加算する報酬関数を，学習の収束先が本来の目的と変わらないように決定する方法を示した．[8] では，例示された動作から報酬関数を生成する方法を提案している．しかし報酬設計の困難さは依然として重要な課題になっている．

その中で分散的な MADRL で協調行動を促す方法の研究も存在する．Tampuu ら [4] は，deep Q-network (DQN) で自律学習する 2 体のエージェントに Pong をプレイさせ，MADRL が協調的な動作を学習できることを示した．更に，球を逃したときの報酬を-1 に固定し，勝利したときの報酬を-1 から 1 の範囲で変動させることで，ラリーの連続回数として表現できるエージェントの協調行動の度合いを変化させられることを確認している．また，Diallo ら [9] は，同様に pong のダブルスの協調行動の学習について論じている．一方，[5] では，global reward を固定し，各エージェントが持つ内在的報酬設計関数を meta-learning の形で学習させている．協調行動が促進されるよう内在的報酬設計関数を学習させることで，global reward だけでは得られない協調行動の獲得に成功している．しかし，あくまで global reward を最大化させる内部報酬を使うため，本研究が対象とするような複雑な構造のタスクには単純に適用できず，特に報酬設計自体への変更を検討する必要がある．

マルチエージェント強化学習で，部分的に分散性を崩して学習を容易化する研究も多数ある．たとえば，Min Tang [10] は，Q テーブルベースの MARL において，他エージェントの視野や経験の活用や，Q テーブルの共有化により，学習速度が速くなることを示した．MADDPG[11] では，Actor-Critic による MADRL において学習時に限り中央集権的な Critic を使い，テスト時には分散したネットワークを使用する手法を提案した．また，DIAL[12] では，エージェント同士の通信を導入し，通信路も含めた勾配を求め policy を更新する手法を提案した．これらのアルゴリズムにより学習時に分散環境では得られない情報を利用できるため性能向上を達成できる．そのため，同期あるいは近接的な行動には効果があるが，本研究が対象とする行動の評価に遅延があり，その遅延も一定では無い場合には，その実装は単純ではない．一方，本研究では報酬設計により分散学習の利点を残したまま協調行動を通して全体の性能向上を目指す．

本研究でベースとした研究 [6] では，自己の部分タスクの達成のみで与える individual reward を徐々に下げ，タス

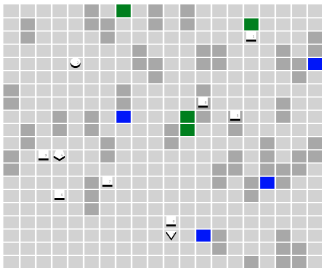


図 1: 環境例

ク全体が終わったときに両方のエージェントに与えられる報酬 (team reward) を徐々に増やすことで，一連の作業を行う 2 体のエージェントの間の協調行動を促進できることを示している．しかし，異なる機能をもつ 2 種のエージェントのみをモデル化しており，より複雑なタスク構造の場合への適用については議論されていなかった．本研究ではこれを拡張し，多段のタスク構造に対する効果を検証する．

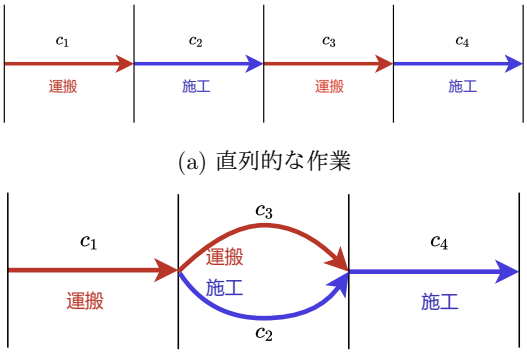
3. 問題設定

3.1 環境のモデル

$A = \{1, \ldots, n\}$ をエージェントの集合とする．本稿では N 種類の異なる機能を備えたエージェントが，$E \times E$ のグリッド環境 V (N, E は共に正の整数) を上下左右に移動しながら指定された場所でそれぞれの部分タスクをある順序で実行する行動を考える．全てのエージェントは離散的な時刻（ステップ）を共有し，1 ステップに 1 マスずつ移動できる．エージェントには観測範囲（視野）があり，自身の周辺 $S \times S$ の情報を獲得できる．ここで正の奇数 S は視野のサイズを表す．環境 V 内には作業場所のマスが $K(> 0)$ 個存在し，各エージェントはそれらのマスへ移動して作業を進めると報酬を得られる．さらに，$P(> 0)$ 種類の資材置き場を表すマスが同数ずつ存在する．作業場所と資材置き場は異なるマスにあり，エージェント同士は同じマスに存在できないが，エージェントは，作業場所やと資材置き場に存在できる．**図 1** に環境の例を示す．灰色のマスは作業場所，白の図形は形ごとに種類の異なるエージェント（ここでは 4 種類），青と緑のマスは資材置き場を表す．

3.2 タスクの構造

タスクの集合を $\mathcal{T}$ と表す．各タスク $T_k \in \mathcal{T}$ は，複数の部分タスク t_k^j の順序付き集合で表し，これを $T_k = \{t_k^1, \ldots, t_k^L\}$ とする．なお，L は部分タスクの数である．ここで部分タスク間に実行の順番を半順序として導入する．このために部分タスク t_k^j はその事前条件 $U_k^j (\subset \{t_k^1, t_k^2, \ldots, t_k^{j-1}\})$ を持ち，U_k^j に含まれる部分タスク全てが実行し完了した後に t_k^j を実行できるとする．ただし，$U_k^1 = \varnothing, U_k^j \neq \varnothing$ $(j \neq 1)$ であり，全てのタスクは部分タスク t_k^1 から開始されるものとする．部分タスク t_k^L が実行された場合をタスクが完了したとみなす．なお上記の条件のもとですべてのタスク



(a) 直列的な作業

(b) 半順序の作業

図 2: タスクの構造の例

T_k 実行可能，つまり事前条件にしたがってすべての部分タスクを実行できる．実際に，U_k^j は，1 から $j-1$ までのインデックスがついた部分タスクの部分集合を条件としているため，これは明らかである．

たとえば，**図 2** は，建築現場でのロボット作業を想定したタスク例であり，**図 2a** は，4 つの部分タスク $t_k^1, \ldots, t_k^4$ を順番に実行するタスク，**図 2b** は，t_k^1 の後に，t_k^2 と t_k^3 は順不同で実行できるが，t_k^4 はそれらが完了してからのみ実行できることを示す．たとえば後者は，$U_k^1 = \varnothing, U_k^2 = \{t_k^1\}$, $U_k^3 = \{t_k^1\}$. $U_k^4 = \{t_k^1, t_k^2, t_k^3\}$ と表現できる．なお，以下では，タスクは一種類のみとして，T_k を T と添字を省くこともある．

エージェント $i \in A$ は各タスクで実行可能な部分タスク t_k^j をもつ．以降，このエージェントを c_j タイプと呼び，c_j タイプのエージェントの集合を C_j と表す．$A = C_1 \cup \cdots \cup$ かつ排反とする．また部分タスクは，対応する資材置き場から作業箇所への資材を運ぶ運搬タスクと，その資材を施工する施工タスクに分かれる．運搬を担うエージェントを運搬エージェント，施工を担うエージェントを施工エージェントと呼ぶ．各エージェントは，各ステップごとに上下左右の隣接するマスへ移動する．行動順はステップごとにシャッフルされる．環境 V の外に移動しようとした場合や，移動先に他のエージェントがいて衝突する場合は，その行動は無効になり現在地に留まる．

運搬エージェントは資材の運搬状態の情報を持つ．運搬中ではない運搬エージェントが，そのエージェントが担う資材の置き場に入ると，そのエージェントは自動的に資材の運搬中になる．運搬エージェントが運搬中かつ自身の部分タスク t_k^j の事前条件が全て満足しかつ t_k^j が未実行のマスに訪れると，部分タスク t_k^j が達成され，資材がその場所に置かれたと考え，資材運搬中ではなくなる．他方，施工エージェントについては，対応する部分タスクの実行条件が満たされ，その部分タスクが未実行である位置へ移動することで，施工が完了するものとする．それ以外の場合には，部分タスクの実行状態は変わらない．

3.3 制限時間

本研究では，協調行動が必要な作業を表現するために，部分タスクが 1 つでも行われたタスクに制限時間を付ける．これは，協調作業が途中で長時間放置されることを防ぐと同時に，実際に我々が想定する建築施工ロボットの，仮止めや接着などの一定時間に完了させる必要がある作業を表現している．そのために，各タスク T_k に完了期限 $w > 0$ を設定し，最初の部分タスクが完了してから最後の部分タスクが完了するまでに ω を越えると実行に失敗（実行はキャンセル）したとして，タスクの状態が初期化され元に戻るものとした．なお，ここで，$\omega = w/(L-1)$ と置く． これに伴いタスク $T_k \in \mathcal{T}$ に対し，期限までの残り時間 v_k を設定する．まず最初の部分タスク t_k^1 が完了すると $v_k = \omega$ とし，以降各ステップごとに k を 1 づつ減少させる．もし，$v_k > 0$ である内に実行可能な部分タスク t_k^j が完了すると，$v_k \leftarrow v_k + \omega$ と増加するが，途中で $v_k = 0$ となるとそのタスクの実行は失敗となる．これにより，各エージェントは，ω 以上の間隔をあけることなく，順次実行することを要求される．実行の失敗は，たとえば，建築作業では資材が無駄になることであり，避けるべき事象となる．また，部分タスク間に時間制限を設定すると，後続のエージェントを考慮しない作業は失敗する上，作業に時間をかけると後続のエージェントの学習機会も減るため，何らかの協調行動が必要となる．

4. 提案手法

4.1 報酬減衰配分法

ここでは，タスク T の完了が目的であり，基本的にはタスク完了時にチーム報酬 (team reward) として，関連する部分タスクを実行したエージェントに報酬 $r(>0)$ を与える．しかし，各エージェントの学習を進めるために，本研究では，学習時には，2 段階で報酬を与えることとする．具体的には，エージェント i が担当できる部分タスクを時刻 t に完了したときに $r_1(t) \geq 0$ の報酬を受け，その部分タスクを含むタスク全体が完了したときに $r_2(t) \geq 0$ を受ける．なお，$r_1(t) + r_2(t) = r$ とする．初期状態で $r_1(0)$ を r の十分大きな割合 (たとえば，$1 \cdot r$, $0.5r$ など) とし，$r_1(t)$ を t の広義単調減少関数とし，最終的に $r_1(t) = 0$ (つまり $r_2(t) = r$) とする．なお，$r_1(t) = 0$ とする t の最小値を報酬境界と呼び，正整数 B_r で表す．

従って，自分が担当する部分タスクが完了しても，最後まで終わらずに制限時間かエピソードの終了に達した場合は最初の報酬 ($r_1(t)$) しか得られない．それにより，学習の初期は自己の作業に集中させて team reward を得る頻度が極端に低い問題に対処し，同時に次のエージェントに学習の機会を与える．その後，$r_2(t)$ を徐々に増やし，協調行動も考慮する学習へと移らせて，最後には自己の部分タスクの行動と協調行動を両立したエージェントを得る．

4.2 Experience Relay Memory の制御

報酬を 2 段階に分けたため，各行為に対する真の報酬は遅れて判明する．そのため，待機用のキュー (experience queue, EQ) を各エージェントに導入し，経験を各エージェントの experience replay memory (ERM) に登録する前に EQ に保存する．EQ のサイズは w もしくはそれ以上とする．

エージェント $i \in A$ が時刻 t に観測した状態を $s_i(t)$，そのときに選んだ行動を $a_i(t)$ とし，その実行で報酬 $R_i(t)$ を受け，状態 $s_i(t+1)$ に遷移したとするとき，これらの 4 データの組 $(s_i(t), a_i(t), R_i(t), s_i(t+1))$ を EQ に保存する．ここで，もし i が自分の部分タスクを実行したのならば $R_i(t) = r_1(t)$ であるが，この場合は，タスクの情報（ID など）を付加する．よって，$(s_i(t), a_i(t), R_i(t), s_i(t+1), T_k)$ と表せる．もし，T_k が完了し，それに関する team reward があれば，対応する部分タスクの実行部分の報酬を $R_i(t) = r$ と置き換える．EQ には各 t ごとにデータが追加されるが，同時に EQ からあふれた 4 データの組（T_k を除く）は ERM に登録する．その結果，部分タスクを実行しても期限までにそのタスクが完了しない場合は，$R_i(t) = r_1(t)$ となり，完了すれば $R_i(t) = r$ として EBM に登録され，これが学習に利用される．

4.3 学習ネットワーク構造と入力情報

各エージェントは，独立した Q 値を学習する深層強化学習のネットワークを持つ．深層強化学習のアルゴリズムには，DDQN[13] を使用する．使用した Q ネットワークの構造を **図 3** に示す．

各エージェントのネットワークへの入力には，global view と local view の 2 種類の観測情報を使用する．global view からは，環境 V の大きさと自己位置を取得できるものとし，縦横のサイズが V と同じ実数値の行列に，自身の移動軌跡を指数的な減衰関数で表す．具体的には，各エピソードの開始時に全要素を 0 で初期化し，ステップ毎に，全要素を 0.9 倍し，最後に自己位置に対応する要素を 1 に置き換える．

他方 local view はエージェントの視野サイズ S を反映し，自身の周辺 $S \times S$ マスの情報を表現した行列であり，各要素（マスに対応）について次の情報を持つ．なお，$S \times S$ が環境 V の外にある部分は 0 で埋める．

- 作業箇所で各部分タスク $t_k^j \in T_k$ について，それがまだ行われていなければ 1，行われていれば 0 を表したベクトル．作業箇所でなければ全ての要素を 0 で埋める．$(S \times S \times L)$
- 作業箇所なら，そのタスクの期限 v_k，その他は 0. $(S \times S \times 1)$
- 資材置き場なら，one-hot 表現による，その種類．存在しなければすべて 0 とする．$(S \times S \times P)$

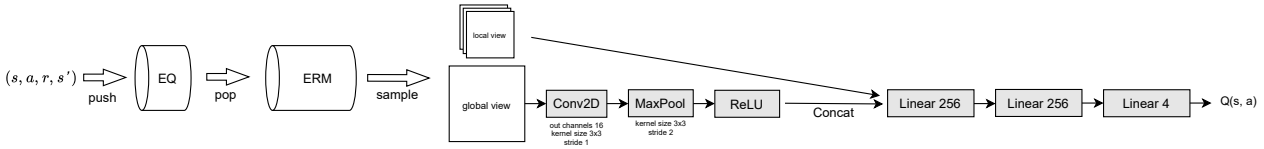


図 3: DDQN ネットワークの構造

表 1: DDQN のパラメータ

設定	値
割引率 γ	0.99
最適化の頻度	4 ステップ
一回の最適化での勾配の更新回数	1
バッチサイズ	32
target network の更新頻度	800 ステップ
ERM のサイズ	120000

- エージェントが存在するなら，エージェントのインデックス i に対応した one-hot 表現． 存在しなければすべて 0 とする．$(S \times S \times n)$
- 運搬エージェントが存在して資材を運んでいるなら 1，そうでなければ 0．$(S \times S \times 1)$

図 3 に示すように，global view を convolutional neural network (CNN) に通してから, local view を結合して multi-layer perceptron (MLP) に入れ，各行動 a に対する Q 値 $\mathrm{Q}(s, a)$ を出力する．

5. 評価実験と考察

5.1 実験設定

環境 V のサイズを 20×20 ($E = 20$)，各エージェントの視野のサイズを $S = 7$ とする．2 種類の施工エージェントが各 2 体ずつ (c_1, c_3 タイプ)，2 種類の運搬エージェントが各 4 体ずつ (c_2, c_4 タイプ) の合計 12 体のエージェントを配置する．施工エージェントは作業場所に行くだけで部分タスクを完了できるのに対し，運搬エージェントは資材置き場と作業場所の両方を訪れる必要があり，作業に時間がかかるため，運搬エージェントをより多く配置した．資材置き場は，各 4 箇所ずつ配置した．

タスクの構造は，直列的な順序の場合（図 2a，実験 1）と，2 番目，3 番目の部分タスクが順序交換可能なやや複雑な場合（図 2b，実験 2）の 2 つを対象とした．1 エピソード長を 1000 ステップとし，エピソード毎に状態を初期化し，エージェントや資材置き場の初期配置もランダムに決定する．V には $K > 0$ 個のタスク $\mathcal{T} = \{T_1, \ldots, T_K\}$ が重ならないようにランダムに配置され，1 つのタスクが完了すると，その場所のタスクは消え，V 内の別の箇所に新たなタスクが 1 つ発生し，タスク数は常に定数 K に保たれる．制限時間のパラメータを $w = 16$ と設定した．$L = 4$ であるため，$\omega = 5$ となる．

表 2: Adam のパラメータ

設定	値
学習率 α	10^{-5}
勾配の指数移動平均の減衰率 β_1	0.9
勾配の 2 乗の指数移動平均の減衰率 β_2	0.999

DDQN のパラメータを **表 1** に示す．探索には ε-greedy 法を使用した．ε は，$t = 1$ から 2,000,000 の間は 1 から 0.05 まで線形に減少させ，$t = 2{,}000{,}000$ 以降は $\varepsilon = 0.05$ で固定する．最適化アルゴリズムには Adam optimizer を使用した．Adam のパラメータを**表 2** に示す．本実験では，1000 万ステップ (10,000 エピソード) まで学習させた．提案する報酬配分法の報酬境界を $B_r = 4{,}000{,}000$ ステップ (4000 エピソード) と設定し，$r_1(0) = 1$ (つまり $r = 1$) から，$r_1(4{,}000{,}000) = 0$ となるように線形に減衰させた．各評価指標は，エピソードの終了毎に，$\varepsilon = 0$ に固定して新しいエピソードを 1 回実行して測定した．

比較手法として，以下の 3 つの固定された報酬配分を採用した．

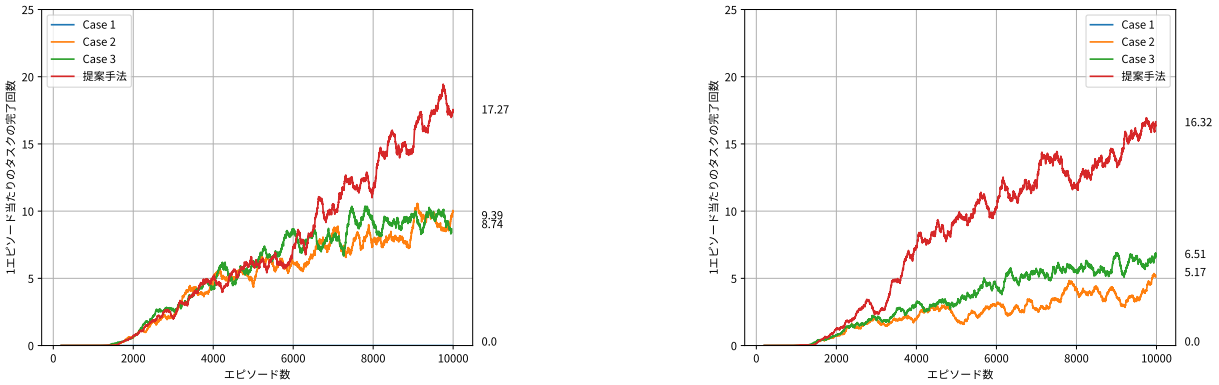
(Case 1) $r_1 = 0,\ r_2 = 1$
(Case 2) $r_1 = 1,\ r_2 = 0$
(Case 3) $r_1 = r_2 = 0.5$

第 1 の配分法は，team reward のみであり，本来タスクが完了して目的が達成できるため自然な設定である． 第 2 は，各部分タスクのみに着目した individual reward のみを学習させ，結果的に全部分タスクの学習を進めることを目指している．第 3 の配分法は，その両方を半分ずつ与え，部分タスクとタスクの完了の両方の達成を目指している．いずれの場合も報酬のスケールを提案手法と合わせて $r_1 + r_2 = r = 1$ が満たされるように設定した．

5.2 効率の比較

実験 1，2 において，提案手法と比較手法の固定的な報酬配分を採用したときの，1 エピソード中に完了したタスク数の推移を**図 4a**，**図 4b** に示す．なお，200 エピソードの移動平均を取っている．

これらの図から，提案手法（赤）が，次に結果の良い比較手法と比べても 2-3 倍程度のタスクを完了させていることが分かる．他方，最終的な team reward のみを与える Case 1 については，(図からはわかりにくいが) どちらの実験でも学習が進まず，完了したタスク数が全ステップを通



(a) 実験 1　　(b) 実験 2

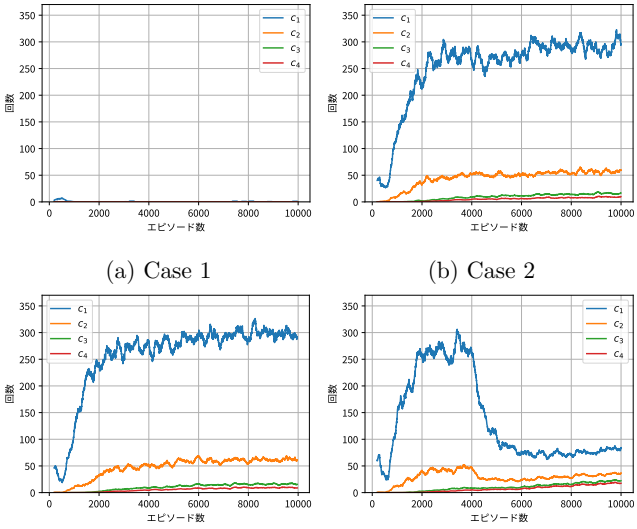
図 4: 1 エピソードの完了タスク数

してほぼ 0 であった．これは，自分の部分タスクを実行しても，報酬が得られず，結果的に全エージェントがランダムな行動に留まったことによるもので，V が team reward だけでは学習が困難な環境であることを示している．

逆に，individual reward のみを与える Case 2（図ではオレンジ色）では，Case 1 よりも完了したタスク数は増えるが，やはりその数は少ない．この場合，team reward がないため，自己の部分タスクのみに着目した行動になる．これは，次のエージェントにも学習の機会を与えることになり，完了するタスクは増えるが，時間制限内に次の部分タスクが完了する割合は少ない．また各エージェントにとっても，報酬獲得の面からは後の部分タスクの完了は考慮する必要はなく，完了したタスク数は頭打ちとなる．この結果は，タスク完了を効率化させるためには部分タスク間で何らかの配慮を取り入れた行動が必要であることを示している．

また individual reward と team reward の両方を与える，Case 3 の場合（緑）では，片方だけを与える場合と比べて完了したタスク数が環境によってはさらに高い．これは，Case 2 と比べて，相互間で何らかの調整行動が獲得したことによる．ただし，自己の部分タスクだけでもある程度報酬を獲得できるため，調整行動が十分に学習されなかったことを示す．

本研究の提案手法（赤）は，それらのいずれよりもタスクの成功回数が高くなった．これは，学習の前半で自己の部分タスクを学習し，その効率を上げるとともに，続くエージェントにも学習の機会を与えたためである．以降，徐々に r_1 を減衰させることで，Case 3 に近い状態となり，前後のエージェントとの調整を含めた行動を学習した．その後，さらに r_1 が減少し，4000 エピソード以降は，team reward のみとなる．このため，自己の部分タスクに加えて，その前後のエージェントとの調整行動がさらに強化され，完了したタスク数（全体の効率）が上昇する．図 4a，図 4b から，4000 エピソード以降も効率が向上していることが分かる．



(a) Case 1　　(b) Case 2

(c) Case 3　　(d) 提案手法

図 5: 部分タスクの完了数: 1 エピソード当たりの，各部分タスクの段階で制限時間に達しタスクがリセットされるか，タスクが最後まで完了した回数（実験 1）

なお，実験 1 と 2 を比較すると，実験 2 のタスク構造の方が完了したタスク数が少ない．これは，順序に曖昧性 (OR 条件) があるために，学習がやや難しかったと思われる．しかし，この場合でも，提案手法ではその差は小さく，効果的であったことが示唆される．

5.3 作業が制限時間に達したときの作業の段階のカウント

各種類のエージェントによる部分タスクの実行状況を把握するために，実験 1 を例に，報酬設定と学習ステップ数ごとに，タスクが制限時間に達したときに，どの段階まで部分タスクが実行されていたかを数えた．この関係を**図 5** に示す．なおここで，たとえば，c_k は，c_k タイプのエージェントの部分タスクまでが完了したことを示し，従って，c_4 はタスクが完了した数でもある．

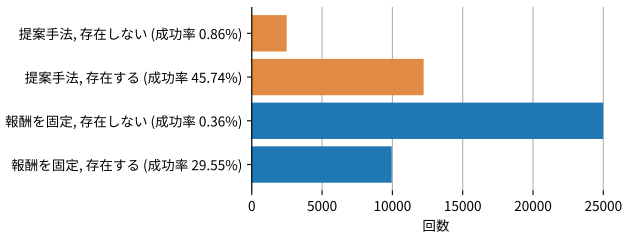


図 6: 視界内次作業エージェント存在数（実験 1）

表 3: 次作業エージェントの視野内存在割合 (実験 1，2)

エージェントタイプ	報酬配分法	存在割合 (実験 1)	存在割合 (実験 2)
c_1	提案手法	75.9%	81.9%
	Case 3	20.0%	11.4%
c_2	提案手法	93.5%	95.6%
	Case 3	56.5%	64.3%
c_3	提案手法	94.5%	95.1%
	Case 3	70.2%	49.4%
平均	提案手法	83.3%	87.6%
	Case 3	28.4%	24.6%

※ 小数点第二位以下を四捨五入した値

まず Case 1（**図 5a**）では，まれに c_1 タイプのエージェントが部分タスクを実行できるが，それだけでは報酬を得られないために，ランダムな行動に終始している．逆に，team reward がない Case 2（**図 5b**）では，それぞれが自分の部分タスクのみの学習に集中し，特に学習機会の多い c_1 タイプのエージェントの作業だけが進み，結果として，大半がそこで時間切れとなってしまう．c_1 タイプの以降のエージェントの学習もある程度は進むが，それは後半に進むにつれて，学習の機会が減り，やはり十分に自分の部分タスクを完了できず，期限切れとなる．Case 3 ($r_1 = r_2 = 0.5$, **図 5c**) でもほぼ，c_1 タイプのエージェントが，自分の部分タスクで得られる individual reward に頼り，Case 2 と同様な結果となるが，team reward も得られるので，ある程度の協調行動により，それ以降のエージェントの学習も進み，若干ではあるが，完了したタスク数が増えている．

これに対し提案手法（**図 5d**）では，前半は，それぞれの部分タスクの学習，特に c_1 c_2 タイプの学習が相対的に進む．その後 4000 エピソード目，つまり r_1 がほぼ消滅する頃から，c_1 c_2 タイプのエージェントの部分タスクの後で期限切れとなるものが急減する．同時に，他の場合と比べ，c_3 c_4 タイプのエージェントまで作業が進む度合いが，他の比較手法と比べて増加する．図 4a と比べても，完了タスク数の増加傾向が長く続く．

5.4 協調行動の解析

提案手法では，4000 エピソード以降，c_1 c_2 タイプのエージェントの部分タスクで期限切れになるものが急激に減る．他のタイプの部分タスク実行数を考慮すると，これはこれら c_1 c_2 タイプのエージェントの部分タスクが完了した回数も減少したと言える．他方で，完了タスク数は増えるので，自己の部分タスクの効率を下げても，全体の完了タスク数を増やすための行動が生成されたと推測できる．それを解析するために，各試行の最後の 100 エピソードについて，エージェントが自己の部分タスクを行ったときに，そのエージェントの視野内に次の部分タスクを行うエージェント間が存在したかを調べた．なお，c_4 タイプのエージェントは後の作業がないため省いた．また，次に作業できるエージェントが複数タイプある場合は，それらすべてのタイプのエージェントが存在する場合のみ，また次のタイプのエージェントが運搬エージェントの場合は，資材を運んでいる場合のみ存在すると判定した．

図 6 に，実験 1 についての結果を示す．橙色の棒グラフは本研究の提案手法，青色の棒グラフは固定配分報酬で最も効率の良かった $r_1 = r_2 = 0.5$ に固定した場合 (Case 3) を表している．参考に，そのタスクが最終的に最後の段階まで完了した割合（成功率）も示した． この実験から，提案手法では，ほとんどが視野内に次のエージェントがいることを確認してから，部分タスクを実行している．またそのときのタスクの完了率も比較手法より高く，続くエージェントが近づいてから部分タスクを実行する（つまり作業箇所に移動する）行動が強化されたと考えられる．他方，比較手法では，2/3 以上で視野内に次のエージェントが存在しない．どちらの手法でも，視野内に次のエージェントが存在しない場合の成功率は極めて少なく，これが連携したタスク間での協調行動と考えられる．

本結果を詳しく調べるために，各タイプのエージェントが部分タスクを実行する際に視野内に続くタイプのエージェントが存在した割合を調べた．これを**表 3** に示す．提案手法に着目すると全ての場合で 75%程度以上，次の部分タスクのエージェントが視野内にある場合のみ，自己の作業を実施している．c_1 タイプではやや低いが，これは次の部分タスクが 2 つあり，今回は両者とも視野内にいる場合をカウントしており，条件が厳しいためと考えられる．他方，報酬配分が固定的な Case 3 では，その比率はどの場合も低い．特に c_1 のエージェントが次のエージェントを無視する傾向が高い．これは，team reward までに長い遅延があり，r_2 の効果を得られにくいからと考えられる．

5.5 考察

図 4a，図 4b より，提案手法による報酬配分法が，他の比較手法による報酬配分法より高い性能を示す行動を学習できることを確認した．特に，その傾向は複雑なタスク構造を持つ実験 2 でも現れ，その差は顕著となる．本研究の手法は複雑な構造をもつ作業を学習させる場合に，相対的に有効であると思われる．

ⓒ 2022 Information Processing Society of Japan

図 5d を詳細に見ると，c_2 タイプのエージェントの学習が c_1 タイプのエージェントと似た傾向を示す一方で，c_3 c_4 タイプのエージェントの学習の進行は遅い．これは，c_2 は施工エージェント，c_3 は運搬エージェントであり，運搬エージェントの作業には時間がかかるためと考えられる．提案手法で r_1 を減少させることにより，c_1 や c_2 タイプのエージェントが次の作業のエージェントの位置を考慮するよう学習し，それが c_3 タイプのエージェントに学習機会を与え成功率を上げた．さらに，それが c_3 タイプのエージェントに続く c_4 タイプのエージェントの学習機会を増加させた．また，提案手法ではシステム全体の性能向上だけでなく，無駄になる部分タスクの実行の回数も減る．これは，今回対象アプリケーションとして考えている建築現場において，建築資源を無駄にする量が減ったことを意味し，望ましい結果と言える．

図 6 より，報酬が固定されていると，次のエージェントを気にせず，自己の部分タスクを遂行することが報酬獲得に有利であると言える．表 3 より，学習が優先的に進む「先に実行するエージェント」が，次に続くエージェントが追いつけない頻度で作業を進める．他方，提案手法では，各エージェントは次の作業のエージェントが視野内に存在するときまで待ち，作業を進める傾向がある．しかし別の観点から見ると，自己の作業効率（実行部分タスク数）が低下することをも意味する．しかし，4000 エピソード以降 $r_1 = 0$ となっても，$r_2 = 1$ の報酬が適切に ERM のデータに反映され，自己の行動と「待つ」という協調行動のバランスを取ったものとなると思われる．

6. まとめ

本研究では，分散的なマルチエージェント深層強化学習を複雑な作業順序のあるタスクの学習に適用でき，自己の部分タスクの学習と協調行動の両者をバランスよく獲得できる報酬配分アルゴリズムを提案した．その手法では，学習時間が増えるにしたがって個人の報酬を減衰させ，作業がすべて終わったときに与える全体の報酬を増加させた．また行動に対する報酬を適切に学習させるために，ERM に加えるデータを 2 段階で保存する点に特徴がある．実験結果から，提案手法が複雑な構造を持つタスクで特に有効であること，エージェントが無駄になる作業を減らせることも確認した．この効率化は，エージェントが作業をするときの周囲の後続エージェントを考慮した協調行動を学習したことが要因であると確認した．我々が対象とした建築現場のアプリケーションを考慮すると，無駄となる作業や資材が減り，提案手法が優れていると言える．

今後の研究として，まず，個々の作業後と作業全体が終わったときの 2 段階だけでなく，その中間でも報酬を与えるアルゴリズムの作成がある．また，報酬の配分比率の学習，報酬の減衰速度の決定の自動化，エージェント毎に個別の報酬配分比率の制御，なども検討する必要がある．これらのパラメータの最適値は各部分タスクの難易度やそれらを含むタスク構造によって異なるため，学習状況等から自動的に決定することが望ましく，より性能の高いアルゴリズムを作れる可能性があると考える．

参考文献

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.: Playing atari with deep reinforcement learning, *arXiv preprint arXiv:1312.5602* (2013).

[2] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P. et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning, *Nature*, Vol. 575, No. 7782, pp. 350–354 (2019).

[3] Yu-Han Chang, Tracey Ho, L. P. K.: All learning is local: Multi-agent learning in global reward games, *Proceedings of Neural Information Processing Systems (NIPS-03)* (2003).

[4] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J. and Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning, *PloS one*, Vol. 12, No. 4, p. e0172395 (2017).

[5] Du, Y., Han, L., Fang, M., Liu, J., Dai, T. and Tao, D.: Liir: Learning individual intrinsic reward in multi-agent reinforcement learning (2019).

[6] Miyashita Y., S. T.: Coordinated Behavior for Sequential Cooperative Task Using Two-Stage Reward Assignment with Decay, *Neural Information Processing. ICONIP 2020. Lecture Notes in Computer Science*, Vol. 12533 (2020).

[7] Ng, A. Y., Harada, D. and Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping, *Icml*, Vol. 99, pp. 278–287 (1999).

[8] Ng, A. Y., Russell, S. J. et al.: Algorithms for inverse reinforcement learning., *Icml*, Vol. 1, p. 2 (2000).

[9] Diallo, E. A. O., Sugiyama, A. and Sugawara, T.: Learning to coordinate with deep reinforcement learning in doubles pong game, *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, IEEE, pp. 14–19 (2017).

[10] Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents, *Proceedings of the tenth international conference on machine learning*, pp. 330–337 (1993).

[11] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P. and Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments, *arXiv preprint arXiv:1706.02275* (2017).

[12] Foerster, J. N., Assael, Y. M., De Freitas, N. and Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning, *arXiv preprint arXiv:1605.06676* (2016).

[13] Van Hasselt, H., Guez, A. and Silver, D.: Deep reinforcement learning with double q-learning, *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30, No. 1 (2016).

ⓒ 2022 Information Processing Society of Japan