東 京 電 機 大 学

博 士 論 文

複数台異種学習ロボットによる知識の共創に関する研究

A Study on Knowledge Co-creation of

Heterogeneous Multi-agent Robot Learning Systems

2015年 3月

河野　仁

目 次

図 目 次

表 目 次

第1章　序論

1.1　マルチエージェントロボットシステムの社会的有用性

近年，家庭環境や公共エリアなどにおける自律ロボットの実世界応用が期待されている [1,2]．たとえば ASIMO や ENON など，アミューズメント利用やショッピングモール内での案内役を想定したロボットシステムである [3,4]．また，Roomba などの家庭用掃除ロボット [5] も普及し始め，他にも介護・生活支援や，レスキューロボットなど様々なロボット技術が研究・開発されている．本論文では，アミューズメント・ホームユースや医療用ロボットなどの共生型ロボットシステム，レスキューロボットや宇宙用ロボットなどの極地活動用ロボットを総称して，サービスロボットと呼ぶこととする．1960 年代から実用化され始めた産業用ロボットは，特定作業への利用を目的とし，技術や市場が成熟し始めている．これに対しサービスロボットは，安全柵でかこまれた産業用ロボットと異なり，我々が生活する様な環境で動作することが前提となる．つまり，より複雑な環境や動的で不確実な環境でも安定した動作が求められる．

また，サービスロボットに求められている機能のほとんどは，人間の作業の代替である．とりわけ，荷物運搬や自律警備システム，被災地探査などの重労働や危険作業を代替するシステムが要求されていると考えられる [6–10]．文献 [6,8,9] などのように，サービスロボットは複数台の自律ロボットにより構成される場合も多い．これは，ある自律ロボットが故障や不調になっても，他の自律ロボットが仕事を代替し，ロバストなシステム構成が可能となるためである．また，1 台のロボットでは可能な仕事に限界がある．複数台のロボットが協調して仕事を行うことで，単体では持ち上げられない重量物の運搬が可能となるなどの利点もある．したがって，複数台の自律ロボットで協調作業を行わせることが，サービスロボットの実世界応用への有効な手段であると考える．以降，複数台の自律ロボットで構成されたロボットシステムをマルチエージェントロボットシステム (Multi-agent robot system: MARS) と呼ぶ．

これまで，MARS による協調作業，協調動作などの研究は多く議論されてきた [11–14]．これらの研究は，実環境である特定のタスクを遂行するには非常に有益な効果を我々にもたらす．しかし，予め行動や計画がプログラムされていると言える MARS は，我々が

生活する様な動的環境において，適応的に行動できるとは言い難い．なぜならば，自律ロボットが行動などの設計段階で考慮されていない環境状態に遭遇した場合，適切に振る舞う保証がないからである．また，全ての環境に適応するプログラムを開発することは，考慮すべき環境状態が多く，設計・開発コストの面でも現実的でないと考えられる．以下に，MARS の課題をまとめる．

- 想定されていない環境においては，自律ロボットは適応的な行動が行えない
- 全ての環境を想定した，プログラミングは現実的でない

環境に対して，自律ロボットが適応的に振る舞うための新たな手法として，自律ロボットが環境状態に対する行動を学習する手法が提案されている [15–18]．とりわけ，強化学習と自律ロボットを融合した研究は，活発に議論が行われている．また，MARS のように多台数の自律ロボットが存在するシステムで強化学習を利用するマルチエージェント強化学習 (Multi-agent reinforcement learning: MARL) も研究されている [19, 20] ．これらの利点としては，自律ロボットに振る舞いを予め組み込む必要がないだけでなく，ハンドコーディングより最適な行動を獲得可能であったり，制御則等を適宜改善出来ることが期待できる．さらに，MARL では協調的な行動も学習可能であり，前述の MARS の課題を解決する手法として期待される．

本論文では，MARS が実環境において適応的に振る舞うための手法として，強化学習により行動を学習する自律ロボットの議論を行う．

1.2 強化学習と MARS の融合

強化学習アルゴリズムが実装された自律ロボットは，環境に対する行動を試行錯誤的に学習する．これにより，予め行動や移動経路をプログラムすることなく，ロボットは環境に対して適応的に振る舞うことが可能となる．また，MARS の各自律ロボットが強化学習を用いることで，協調的な行動を獲得することも可能である．

これまで基礎的な検討として，実ロボットを用いないエージェントベースでの研究では，荒井 [21–23] らの研究や Tan の研究 [24] がある．これらの研究は，ハンターエージェントが獲物を捕まえる行動を学習する追跡問題をベンチマークとして採用し，いくつかの強化学習器の効果や，センサ情報や知識の共有の学習への効果，ハンターエージェントにおける協調作業獲得の効果等を示した．

また，MARS と強化学習を融合した代表的な研究に，Matarić [25] の研究がある．Matarić [25] は，複数台ロボットが存在する動的環境における，複数台強化学習自律ロボットによ

る強化学習を，Heterogeneous reinforcement function と Progress estimation の導入で実現している [26]．保田 [27] は，強化学習器の連続空間への適応，頑健性の増大を課題とし，Bayesian-discrimination-function-based reinforcement learning (BRL) を用いて，マルチロボット強化学習システムにおける協調搬送タスクや協調荷上げタスクへの応用を行った．保田の研究では，環境変化検知や環境のダイナミクス低減など，強化学習の MARS 応用に関連する諸問題にもアプローチを行い，成果を得ている．福田らは，学習の効率化を目的に，エージェントの自律的な環境変化検出のための多重強化学習法を提案している [28]．これらの研究だけでなく，Yang ら [19] や Graña ら [20] のサーベイからもわかるとおり，過去十数年にわたり様々な MARS と強化学習の融合，さらには他機能の融合が研究されてきた．

これらの MARS と強化学習を融合した，マルチロボット強化学習においては，強化学習だけでなく，情報共有機能や環境変化検出機能などの周辺機能も統合することが重要であると考えられる．すなわち，マルチロボット強化学習システムとして構成することが，実世界応用においては重要であると言える [23]．

しかし，荒井は文献 [23] において，MARL はシングルエージェントにおける強化学習の諸問題を継承すると指摘している．同様に，MARL の具体系であるマルチロボット強化学習において，利点を継承しつつもいくつかの課題も残されていると考えられる．以下にマルチロボット強化学習の課題をまとめる．

(1) 学習に長時間，もしくは多くの繰り返しが必要
(2) ロボットは遭遇した環境に対する振る舞いのみを学習
(3) 獲得した知識の記憶量の限界

上記課題 (1) は，強化学習アルゴリズムに起因している．課題 (2) は，直感的には必然であるが，強化学習エージェントやロボットが新たな環境に遭遇する度に，長時間・多くの繰り返しを行い環境への振る舞いを学習する．これは，環境に対する即応性が無いと考えられる．課題 (3) は，ロボットにおけるハードウェアリソースが有限であることも原因であるが，なにより，伝統的な強化学習では考慮すべき環境状態変数が増加すると，指数関数的に環境状態数が増大する，「**次元の呪い**」が発生することも原因である [29]．また，仮に今後のコンピュータ技術発展に伴い，膨大なメモリ空間が確保できたとしても，全ての環境状態を保存し，時間をかけて学習をすれば容易とは考えにくい．これは，ノーフリーランチ定理のようにに，全ての評価関数に有効な解は存在しないという立場をとる [30]．それぞれの環境に適した解法（知識）を用意・保存し，適宜使い分けることが重要であると著者は考える．そこで，知識の共有と保存，知識の再利用など，既に獲得

した知識をロボットが活用する手法が上述の問題点に対して効果的であると考えられる．知識の共有と保存，再利用が可能となれば，強化学習エージェントやロボットは，始めは長時間・繰り返し，学習を行わなければならないが，一度学習してしまえば似たような環境に遭遇した場合，過去の知識を活用できる．また，様々なタスクを多くの強化学習エージェントやロボットが知識を獲得し保存すれば，あるエージェントが初めて遭遇する環境においても，知識を適宜共有・選択し即時的に行動を行うことが可能であると期待できる．もちろん，これらのようなロボットが知識を活用する研究は成されており，次節から従来研究を述べる．

ここで，本論文におけるエージェントやロボット，知識の用語を定義しておく．エージェントは広く意味を包括する用語であり，しばしば実ロボットのことをエージェントと記す論文がある．本論文においても特に断りが無い場合，実世界に存在するエージェントであるロボットや仮想的なエージェントプログラムを，エージェントと統一して呼ぶ．しかし，**強化学習ロボット**は，強化学習アルゴリズムが実装された，身体性のある実ロボットのことを示し，複数台の強化学習ロボットは**マルチロボット強化学習**とする．また，**強化学習エージェント**は強化学習アルゴリズムが実装された，計算機内の仮想的なエージェントプログラムを示すこととする．複数台の強化学習エージェントは**マルチエージェント強化学習**，または MARL と呼ぶ．

1.3 知識の活用に関する従来研究

これまでの強化学習研究において，過去にエージェントが獲得した知識を活用する試みが行われている．これらの研究を俯瞰すると,「共有」や「保存」,「再利用」という 3 つの方法に分類できる (図 1.1)．それぞれが完全に独立した方法ではなく，お互いの手法が補完し合っている場合もあることに注意が必要である．本稿では，3 つの分類を用いて従来研究を概説する．

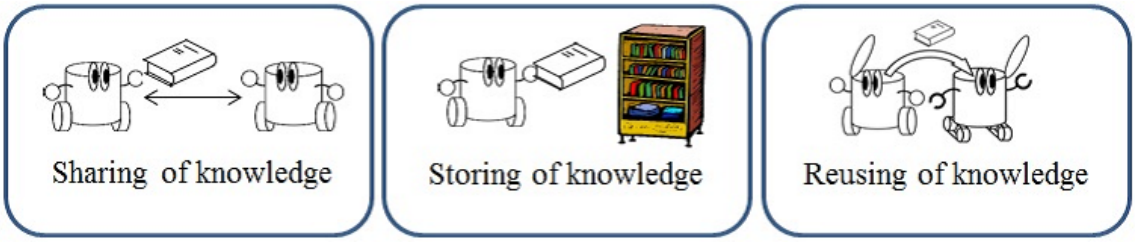


図 1.1 知識の共有・保存・再利用

1.3.1 知識の共有

複数台の強化学習エージェントが知識を共有する効果について明らかにした研究では，Tan [24] の研究が先駆的である．この研究では，追跡問題をベンチマークとして採用し，複数台のハンターエージェントが観測情報を共有したり，方策 (本研究の知識) を共有することにより，学習の高速化と MARL の協調行動獲得に関して効果を発揮している．しかし，Tan は情報を共有しないマルチエージェントと，情報を共有するマルチエージェントでの学習速度の違いを考察しているだけであり，方法論の確立には至っていない．また，計算機実験の環境はホモジーニアスな強化学習エージェントである．実世界での応用を考慮すると，ヘテロジーニアスなマルチエージェントにおける議論が必要であると考える．

Ahmadabadi ら [31,32] は，MARL において，方策（本研究の知識）を通信により共有し，他エージェントが獲得した知識を自エージェントに取り込む Weighted strategy sharing (WSS) architecture を提案している．しかし，Ahmadabadi らの研究では，知識を共有するエージェント同士の経験が同様であると効果が無く，熟練度の異なるエージェント間において，WSS の効果が現れたと述べている．また，強化学習エージェントはホモジーニアスに設定されている．

1.3.2 知識の保存

知識の保存に関するアプローチとしては，例えば，知識の再利用が前提であるが保田らの研究がある [33]．保田らは，Instance-based classifier generator (IBCG) という強化学習法で獲得した知識を，Naive bayes model の形式で保存し，強化学習ロボットが環境変化を検出した際に，適宜知識を選択・利用するシステムを実現している．マルチロボット強化学習への適用は行われていないが，実験結果から環境変化が伴う環境においても頑健に振る舞うことを確認したと述べている．

さらに特に近年は，計算速度の向上，情報記憶技術の高密度化が進み，クラウドコンピューティングとロボットシステムを融合したクラウドロボティクスが提唱されている [34]．クラウドロボティクスの代表的な研究として，RoboEarth がある [35–37]．RoboEarth は，自律ロボットのクラウドコンピューティングリソースへのアクセスを容易にするフレームワークである．特に，文献 [37] では，強化学習ロボットがクラウドを用いて知識を共有するデモンストレーションを行っている．しかし，コンセプトとして強化学習の知識共有を掲げているが，方法の確立には至っていない．

1.3.3 知識の再利用

獲得知識の共有とは異なり，知識の再利用をするアプローチも行われている．このアプローチは，Tan [24] や Ahmadabadi ら [32] の研究と異なり，シームレスに知識を共有するのではなく，時系列的に過去の知識を利用し，現在の学習を高速化する手法である．

松井ら [38,39] は，概念学習を提案し，過去のいくつもの知識から帰納的に政策事前条件という概念を獲得し，新たな環境での学習では，政策事前条件から部分的に政策（本論文での知識）の改善を行い，学習の高速化に貢献する研究を行った．

近年のアプローチとして，Taylor らの転移学習が挙げられる [40]．強化学習における転移学習では，Source task にて強化学習エージェントが知識を獲得し，Target task の強化学習エージェントが Inter-task mapping (ITM) という処理を通して知識の再利用を行う．ITM は，Source task のエージェントと Target task におけるエージェントのセンサ情報と行動のマッピングを行い，知識の再利用を実現するフレームワークである．様々なタスクでの有用性を評価している．Boutsioukis らは，マルチエージェント環境に置ける転移学習の効果を検証している [41]．さらに，Taylor らは学習の高速化を目的としている転移学習を，MARL の各エージェント間で同時並列的に転移学習を行い，さらなる学習速度向上を検討している [42].

知識の再利用という，概念的には新しくない方法論が Taylor らの研究 [40] から議論が盛んに行われ，方法論の確立がなされつつある．

1.4 本論文における議論点

これらの従来研究は，知識の活用に関して重要な要素技術の確立や実験を行っている．しかし，以下に挙げる検討事項に関しては，まだ議論が深く行われておらず，既存研究だけではマルチロボット強化学習の実世界応用が行えないと考えられる．

(1) ヘテロジーニティに関する議論

従来研究では，ホモジーニアスなマルチエージェントシステムであると仮定されることが多い．しかし，実際のマルチロボット強化学習のシステム構成として，完全に同一のロボットを複数台用意することは難しい．また，MARS としては各強化学習ロボットに役割や特徴を持たせる場合もあり，マルチロボット強化学習のシステム構成として，ヘテロジーニアスであることを前提とする必要がある．

(2) **不確定エージェントにおける転移学習**

従来の研究において，強化学習エージェントやロボットは構造や特徴が既知であり，固定的であった．実世界における MARS の運用を考えると，ロボットの故障に伴うバージョンアップや，新型ロボットの購入やリプレースなど，新たなプラットフォームがシステムに参加することは想像に難くない．したがって，強化学習エージェントやロボットのヘテロジーニティは，不確定である，言い換えると新たなエージェントの参加を許容することを前提として議論することが重要である．

(3) **獲得知識の処理に関する議論**

これまでの研究から，強化学習エージェントやロボット同士の知識の共有は効果が確認されていると言える．しかし，これらの研究では予め決められた範囲での強化学習エージェントや強化学習ロボット間での知識共有しか検討されていない．これらは基礎的検討として，重要な設定であるが，インターネット技術や通信技術が発展した現代において，時空間的に離れたマルチロボット強化学習システム同士が知識を共有する，という運用が成される可能性が非常に高い．強化学習エージェントが獲得した知識を「だれが」「どこで」「どのように」保存・管理行うのか議論を行う．

1.2 節でも指摘したとおり，強化学習と自律ロボットの融合だけでなく，周辺機能も統合しマルチロボット強化学習システムとして構成することが重要である．しかし，従来研究だけでは上記検討事項を満足することは無く，従来研究の統合や新たな機能の追加が必要であると考える．したがって，複数台かつヘテロジーニアスを前提とした強化学習エージェントやロボットが，知識を共有・保存・再利用するサイクルを繰り返し，時空間的に異なる強化学習エージェントやロボットが知識を共に創る「**共創**」の概念を導入することを提案する．そこで，本研究では知識活用に関す新たな枠組みとして，知識の共有や保存，再利用，さらに収束判定機能などの周辺機能も統合した「**知識共創**」と，そのフレームワークの提案・検討を行う．

1.5 獲得知識の共創

本論文では，知識の共有と保存，再利用を発展させ，さらにそれらを包括する知識の活用の新たな枠組みとして，知識の共創（知識共創）を提案する．知識共創は複数台へ

テロジーニアスなマルチロボット強化学習における知識の相互運用性 (Interoperability) を高める手法であるといえる．

また，本節では共創と近いパラダイムである創発に関しても触れ，共創と創発の違い，さらに知識共創の意義を明らかにする．

1.5.1 社会における知識共創

共創 (Co-creation) という概念は，一意に定義することはできないが上田らの共創工学を参考にすれば，ある意思決定問題に対して，様々な領域の人々が知を集結し，新たなソリューションを創造する活動や過程であると言える [43]．

知識共創 (Knowledge co-creation) も，新しい概念ではなく様々な分野における，人々の知識の蓄積や創作活動のことを示す．例えば，オンラインコミュニティにおける知識共創 [44] や，新商品開発における消費者知識運用フレームワーク [45] などがある．

金野らの論文 [46] では，“知識共創は人々の間で新しい価値を持った情報を生み出し，伝え合うことであると考えられる” と述べている．有用な知識を必要とする人々が，ソサイエティやコミュニティにおいて，お互いの知識を共有・活用し，相互に作用しながら，より良い知識を作り出すプロセスやモデルを示すと考えられる．知識共創のイメージを図 1.2 に示す．コミュニティに属する人々が知識を共創し，さらにコミュニティ同士，すなわちソサイエティ内でも知識共創が可能である．さらに，既に蓄積された知識を基に，新たに人々やコミュニティが知識を共創することも考えられる．図 1.2 中の Interaction は媒体が必要である．例えば紙媒体，言語などがあり，近年の例で言えばインターネットや，さらに上位の SNS (Social network service) などのシステム的なコミュニケーションも考えられる．

一般的な用語として，知識共創は人々による活動を示す．文献 [43] で述べてられているように，知的な人工物による共創も可能であると考えられるため，知識を必要とする強化学習エージェントやロボットにおいても，知識共創が可能である．

1.5.2 創発と共創

人工知能やロボット工学などの学術分野では，以前から創発 (Emergence) という概念の議論が行われている [47]．自律エージェントにおける創発を例にとると，マルチエージェント環境で基本的な個の振る舞いから予想しない，もしくは複雑な現象（協調的な行動など）が発現することである．創発は，蟻の集団行動や神経細胞による知能の発現

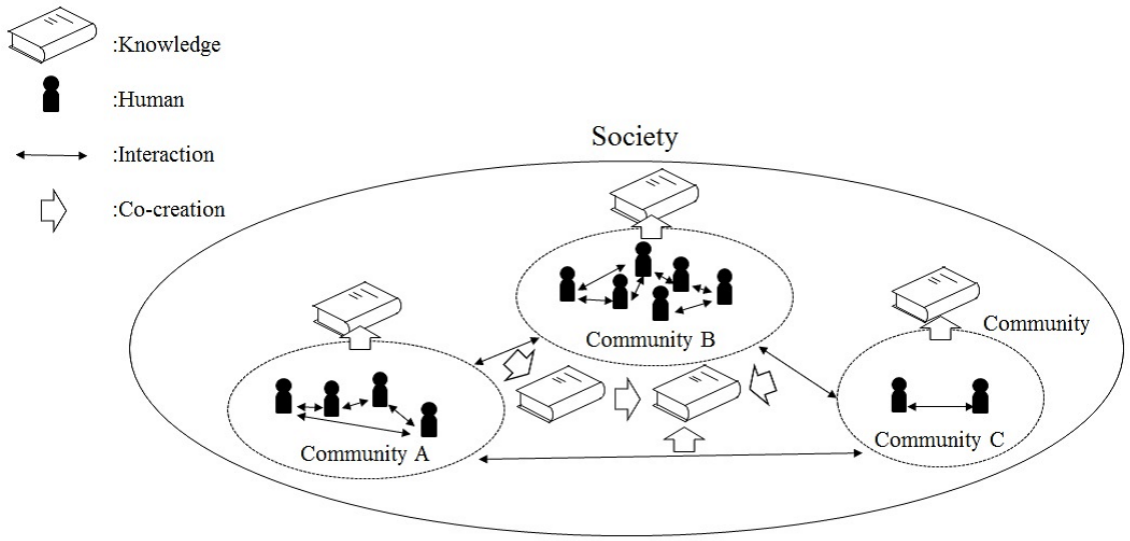


図 1.2 社会における知識共創のイメージ

などをボトムアップ的に構成し，トップダウン的に大域的な挙動の拘束をする（図 1.3）．この双方向の過程を通じて，新たな機能や秩序が形成されることである．構成論的アプローチとも呼ばれる．

創発と共創は非常に近い言葉であるが，創発はメカニズムを構成・解明し，工学的な応用や理学的，生物学的な知見に生かす体系である．すなわち，対象としている系が生物や人間社会，人間–自然, 人間–機械, 機械–機械間相互作用などと幅広い．共創も同様であり，それらの系からソリューションの発現を期待する．

これに対し，本論文で提案する知識共創は，ボトムアップ的に確立されてきた手法を活用し，有用な知識をエージェントが創り出すメカニズムを確立する．また，基本的に人間は介在しない Machine to machine (M2M) の系を前提とする．しかし，創発という幅広い領域において，計算機やエージェントを用いて創発を実現する記号創発がある．記号創発では，知識の創発を目指した研究もおこなわれており [48]，共創は創発と多くの共通項を持っていると考えられる．

したがって，知識共創は共創の部分領域であると同時に，創発的な機能も有する．しかし，本研究における知識共創の恩恵を受けられる者（物）は，知識を作り出した者または知識を利用する者を前提とし，本研究では強化学習エージェントやロボットが対象となる．共創のように新たなソリューションを創造する作業と，創発ほど大域的な挙動の発現を期待しないという所から，知識共創は図 1.4 のような位置に属する．

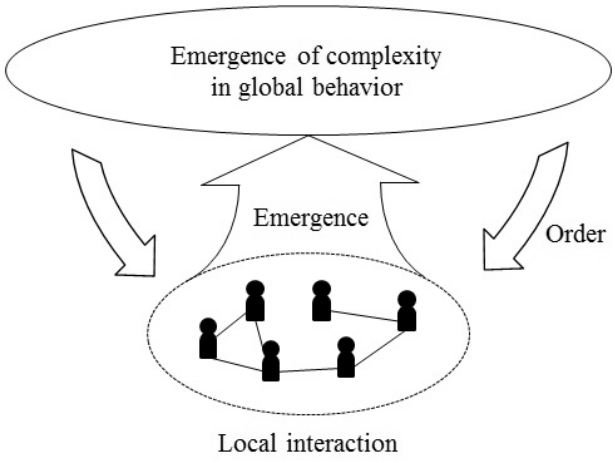


図 1.3 創発のイメージ

1.5.3 マルチロボット強化学習における知識共創

これまで述べた知識共創という活動に着目し，本研究ではマルチロボット強化学習にける知識共創を提案する．これは，時空間的に分散した強化学習ロボットが，獲得知識の共有や知識の再利用，さらに，再利用時に再学習した知識を再度共有し，他の強化学習ロボットや強化学習エージェントに再利用される，というサイクルを繰り返すシステムである．このとき強化学習ロボットは，ヘテロジーニアスであることを前提とし，また再利用する知識の選択や，新たな知識獲得の判定を自律的に行う．

本研究で提案する知識の共創を実現するシステムを，**知識共創フレームワーク**と呼ぶこととする．知識共創フレームワークのイメージを図 1.5 に示す．

知識共創フレームワークにより生成される知識は，膨大な量となることが考えられる．1.2 節の課題 3 で指摘した通り，強化学習ロボットは本体（搭載計算機）の制約からリソースが有限であることから知識の保存に関しては近年議論が活発である**クラウドコンピューティング**（以下クラウド）を利用する．クラウドもリソースは有限であるが，システムの前提としてスケールアウトが容易であり，またデータセンタのような大規模計算機施設を世界各地に設置・連携可能ということから，強化学習ロボットの計算機に対して広大なリソースを利用することが可能である．クラウドは，知識の保存だけでなく，類似知識の統合や不要な知識の削除などを実行する．知識の保存に必要な大規模ストレージや知識の統合などの高次元処理を，計算機リソースが有限である強化学習エージェントやロボットの代わりに実行することが主な役割である．

知識共創システムに参加する各強化学習エージェントやロボットは，クラウドに接続される．クラウドに接続された強化学習ロボットは，他のマルチロボット強化学習シス

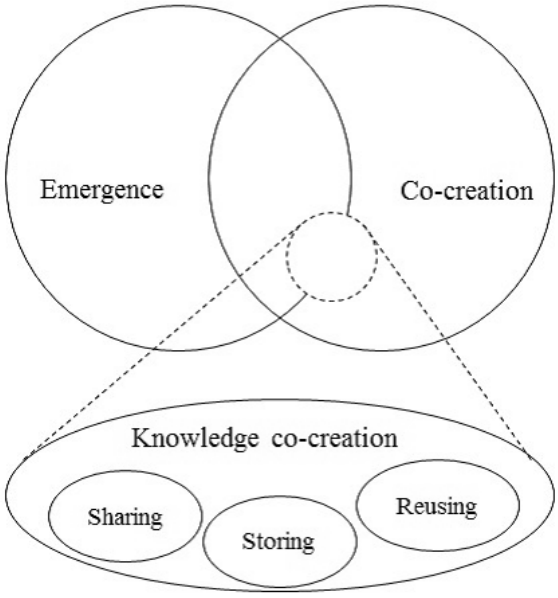


図 1.4 共創と創発，知識共創

テムに属している強化学習ロボットの知識を再利用可能である．もちろん，強化学習ロボットだけでなく，強化学習エージェントの知識の再利用が可能であるとする．

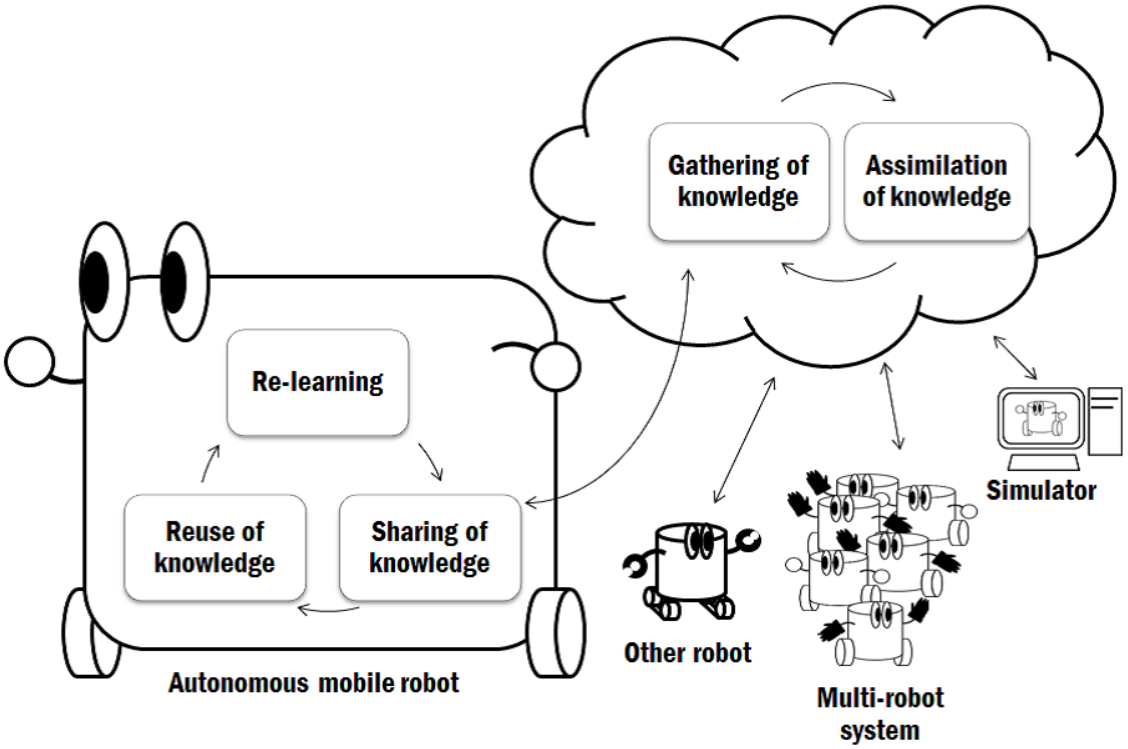


図 1.5 マルチロボット強化学習における知識共創フレームワーク

1.6 本論文の目的

本論文では，マルチロボット強化学習システムや MARL システムが，獲得した知識の共創を実現するために，クラウドを用いた知識共創フレームワークを開発する．知識共創を実現するためには，これまで研究されてきたマルチロボット強化学習や MARL だけでなく，知識の共有や保存，再利用などの機能を始め，知識の獲得判定機能や高負荷・高次元計算を担うクラウドとの連携機能をフレームワークとして構築することが重要であり，以下に示すフレームワークを構成する各要素について開発を行う．

(1) ヘテロジーニアスなマルチロボット強化学習・MARL システムにおいて，知識の再利用を実現する手法
(2) 再利用可能な知識を “どのように” 再利用するのか決定する手法
(3) 自律的な知識共創を行うための，強化学習の知識獲得を自動判定する手法
(4) クラウドを用いた強化学習による獲得知識の統合手法

これらの手法を実現する過程で，ヘテロジーニアスを前提とした，多台数のマルチロボット強化学習における知識共創について議論を行う．また，次節から知識共創フレームワークに必要な機能の全体を俯瞰し，既存技術の利用と本論文にて開発する要素技術を述べる．

1.7 知識共創フレームワークに必要となる機能

知識共創フレームワークには，知識処理をするクラウドや強化学習ロボット，強化学習エージェントが必要である．これらの各要素に必要な役割や内部機能を本節に記す．

1.7.1 前提条件

まず，知識共創フレームワークにおける前提条件を明らかにする．本論文では，以下の機能に関しては既存の技術や手法を用いるため，議論の対象としない．

1.7.1.1 想定エージェント

実世界における MARS は，本章の冒頭で述べたとおり様々なエージェントが用いられる可能性がある．しかし，例えば車輪型ロボットの知識を人型ロボットが再利用するのは容易なことではない．全てのロボットやエージェントを考慮することは現実的でない．

本論文では基礎的な議論として，車輪やクローラにより地面を移動する機構を有する，移動ロボットをエージェントの前提とする．

1.7.1.2 通信インフラ

本論文におけるエージェントは，無線通信インフラを通じてクラウドなどの外部計算機へ自由にアクセス可能であるとする．

実問題では，通信範囲やスループット，プロトコルなど実装段階において考慮しなければならないことが多い．しかし，本論文で提案する知識共創フレームワークは，まだ基礎的検討である．したがって，通信インフラに関する課題やアプローチは今後の課題とし，本論では取り扱わない．

1.7.1.3 タスクの割り当て

本論文において，強化学習ロボットやエージェントが遂行するタスクは，そのシステムの利用者が設定をするものとする．強化学習の報酬設計問題に関連する前提である．強化学習において，タスクの達成は報酬により判断し，環境から与えられるものとする．

したがって，MARS へのタスクの割り当てや報酬設計に関する課題やアプローチは今後の課題とし，本論では取り扱わない．

1.7.2 エージェントの役割と機能

強化学習ロボットは，実世界において与えられたタスクを遂行する役割が必要であると同時に，学習を行い知識を獲得し，強化学習エージェントやクラウドとも連携する必要がある．

また，強化学習エージェントは，強化学習ロボットと異なり，計算機内で仮想的に存在することとなる．必要な機能としては強化学習ロボットと同様であると考えられるが，強化学習エージェントの強みは，実世界とのインタラクションを行わず，計算機リソースを活用して学習が行えることである．つまり，計算機シミュレーションにより知識獲得を行う機能が重要である．

(1) 強化学習機能
(2) 知識の再利用機能
(3) 再学習機能

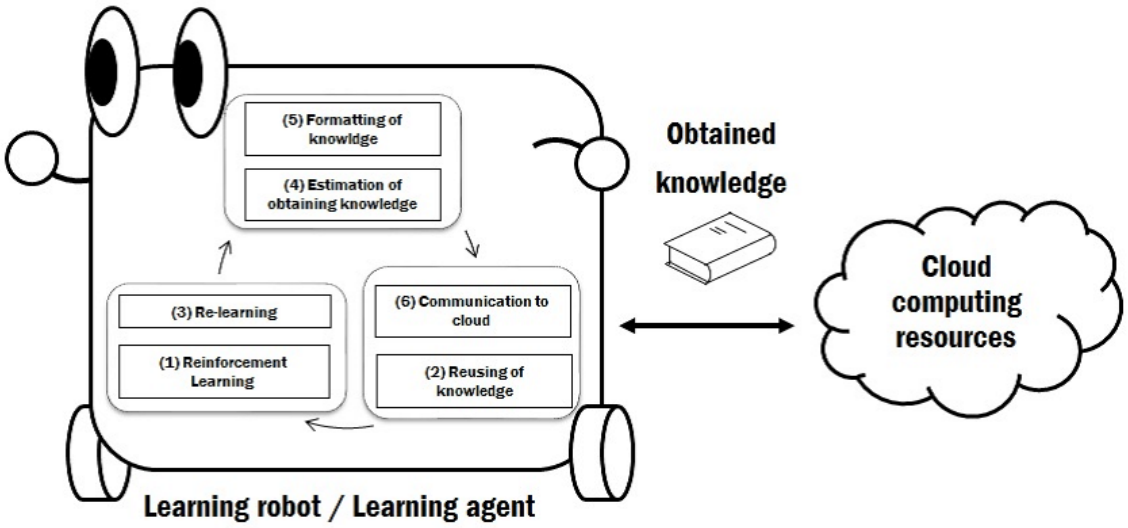


図 1.6 強化学習ロボットやエージェントの必要機能

(4)　知識の獲得判定機能
(5)　知識の保存機能
(6)　クラウドとの通信機能

上記各機能の関係や流れの概要を図 1.6 に示す．また，各要素を以下に概説し，既存技術により対応可能か明らかにする．

図 1.6 において，(1) 強化学習機能は，これまで提案されてきた強化学習アルゴリズムのことであり，既存の強化学習を用いる．(2) 知識の再利用機能は，転移学習のような機能であるがヘテロジーニアスなエージェント間においても知識の再利用が可能なメカニズムが必要である．(3) 再学習機能は，再利用する知識を (1) 強化学習機能で利用することである．共有されている知識を再利用できるように調整する機能が必要である．(4) 知識の獲得判定機能は，中途半端に学習した知識を共有しても効果的でないことは明らかなため，エージェントが自律的に学習による知識の獲得を判定する機能が必要である．その後，強化学習により獲得した知識を保存し，共有できるような形式にする機能が (5) 知識の保存機能である．さらに，エージェントの内部に保存された知識をクラウドにアップロードし，他のエージェントと共有する為の通信を行う機能が (6) クラウドとの通信機能である．

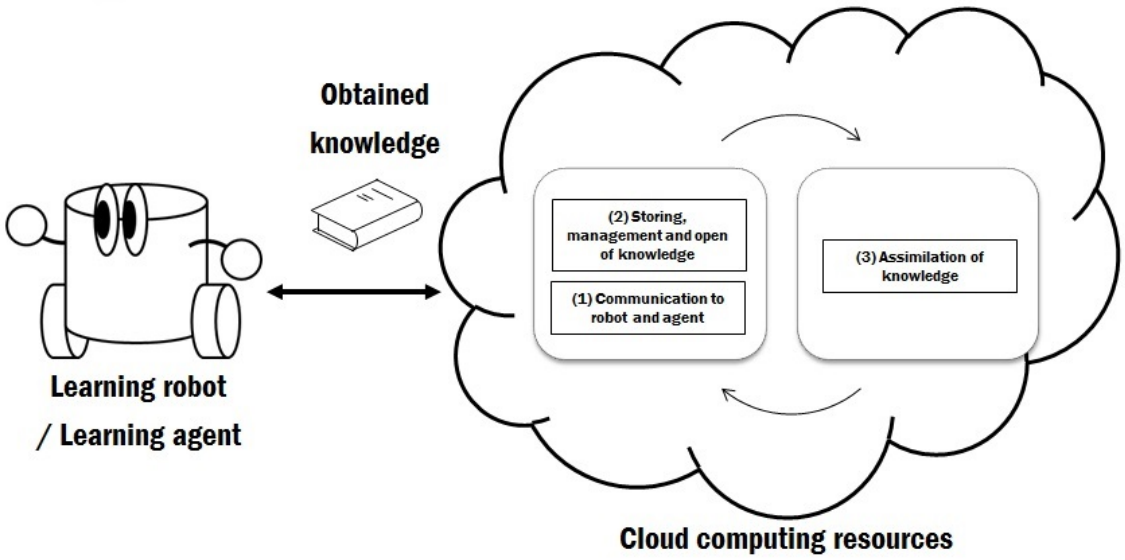


図 1.7 クラウドの必要機能

1.7.3 クラウドの役割と機能

クラウドは，強化学習エージェントやロボットにおいて高負荷な処理を代行する能力に着目し，以下の機能が必要である．

(1) 強化学習エージェントやロボットとの通信機能
(2) 獲得知識の保存・管理・公開機能
(3) 獲得知識の統合機能

上記各機能の関係や流れの概要を図 1.7 に示す．また，各要素を以下に概説し，既存技術により対応可能か明らかにする．

図 1.7 において，(1) 強化学習エージェントやロボットとの通信機能は，ロボットと通信するインフラを始め，知識データを送受信するプロトコルである．インフラは，汎用性の面から無線 LAN を利用できると考えられるが，知識の送受信におけるプロトコルは開発が必要である．(2) 獲得知識の保存・管理・公開機能は，エージェントから受信した知識をクラウド上に保存し，それらをエージェント別やタスク別などで分類して管理する機能である．また，それらの保存した知識を選択し，他のエージェントに送信する機能も必要である．これらの機能は，保存管理は既存技術の DB (Data base) や RDBMS(Relational data base management system)，クラウドで用いられているフレームワークなどの活用により可能であるが，エージェントの要請に応じて，知識を選択する機能は開発が必要である．(3) 獲得知識の統合機能は，エージェントから受信した知識を統合し，データ容

量を削減する機能である．また，知識の統合により保存された知識を整理し，選択しやすくする狙いもある．これらの機能は，基本としてクラウドで用いられているフレームワークが有用であるが，カスタマイズが必要である．

1.8 本論文の構成

本論文は” 複数台異種学習ロボットによる知識の共創に関する研究” と題し，7 章から構成されている．本章では，関連研究，研究の背景と目的，提案手法の概要も述べ，議論点を明らかにした．

第 2 章 強化学習と知識の再利用 では，まず知識共創フレームワークを議論するための強化学習や転移学習などの予備的な知識の説明を行う．また，特に関連性のある従来研究においては，深く議論を行い，本論文における課題を明確にする．

第 3 章 Hetero-MARL における転移学習法 では，ヘテロジーニアスなマルチロボット強化学習において，知識を再利用する為の方法論を議論する．本章では，新たにオントロジを用いた階層的転移学習を手法として提案し，追跡問題を用いた計算機実験から本手法の有用性を示す．

第 4 章 知識の再利用時における転移率の検討 では，強化学習エージェントや強化学習ロボットが知識を再利用する時に，過去に獲得された知識がどれくらい現在のタスクに貢献できるか議論を行う．本章では，転移率による知識の再利用度合の調整パラメータを提案し，タスクの類似度や，ロボットやエージェントのヘテロジーニティに応じて，再利用度合を変更する手法を提案する．また，静的環境や動的環境による追跡問題を用いた計算機実験から転移率の有用性を示す．

第 5 章 強化学習における自律的収束推定法 では，自律的に知識共有を行うために，強化学習ロボットが自律的に知識の獲得を判定する方法論に関して議論を行う．本章では，強化学習の知識獲得の指標である学習の収束に着目し，学習収束推定法を提案する．また，追跡問題による計算機実験により，強化学習エージェントやロボットが自分で学習の収束を推定し，知識を共有する為の手法としての有用性を示す．

第 6 章 外部計算機資源を用いた獲得知識の統合 では，強化学習エージェントやロボットが獲得した知識を，クラウドコンピューティングを活用して知識処理するための方法論に関して議論を行う．本章では，単純に知識の単純和を取るだけでなく信頼度パラメータを導入し，新たに提案した知識結合法を計算機実験により有用性を示す．

第 7 章 結論 として，知識共創フレームワークを実現する 4 つの提案手法をまとめ，本論文の成果を述べる．また，本研究における今後の課題と展望を述べる．

また，1.4 節に提示した本論文の 3 つの議論点と，本論文の提案手法である第 3 章から第 6 章は，以下のように対応している．また，知識共創フレームワークに対して各章の提案手法と既存手法は，図 1.8 のように対応している．

表 1.1 本論文の構成

議論点	＜章＞提案手法
(1) ヘテロジーニティに関する議論	<第 3 章> Hetero-MARL における転移学習法 <第 4 章> 知識の再利用時における転移率の検討
(2) 不確定エージェントにおける転移学習	<第 3 章> Hetero-MARL における転移学習法
(3) 獲得知識の処理に関する議論	<第 5 章> 強化学習における自律的収束推定法 <第 6 章> 外部計算機資源を用いた獲得知識の統合

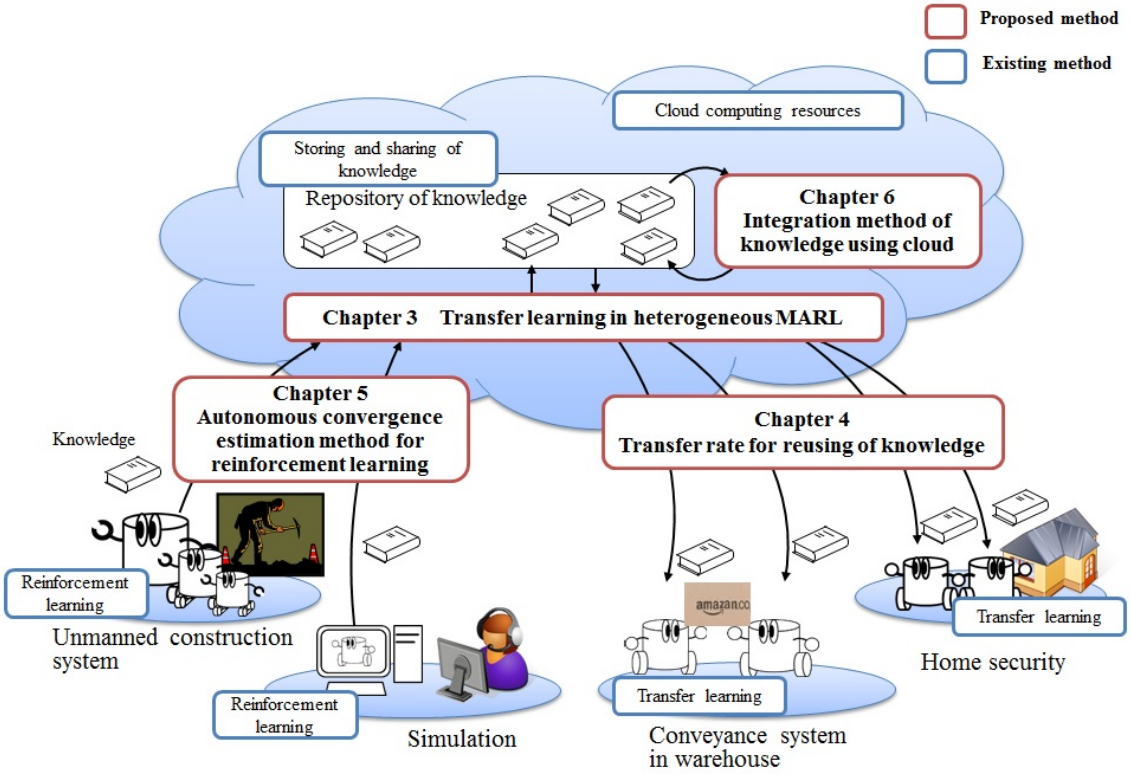


図 1.8 各章の提案手法と知識共創フレームワークの対応イメージ

第2章　強化学習と知識の再利用

2.1　緒言

本章では，マルチロボット強化学習における知識共創の議論に先だち，強化学習や転移学習，その他の関連技術・関連研究に関して述べる．

2.2　強化学習の背景

強化学習は 1980 年代から議論が行われてきた，機械的な学習手法であり，仕事の達成と同時に得られる報酬から，エージェントが試行錯誤的に最適な行動を獲得するアルゴリズムである．Artificial neural network (ANN) など，教師あり学習と異なり教師信号（教えるべき正解）が必要なく，強化学習は探査により解を求めることが特徴的である．強化学習は，モデルとして図 2.1 に示すエージェントと環境のインタラクションが必要である．

強化学習エージェントは環境から環境状態 (State) を観測し，強化学習アルゴリズムにより行動 (Action) を選択し実行する．行動した結果として，環境からは報酬 (Reward) が与えられ，同時に環境の次状態も観測可能である．この一連の流れを繰り返し，強化学習エージェントは環境に対して最適な振る舞いを試行錯誤的に獲得していく．

これまで，様々な強化学習法（例えば Q 学習や TD 学習，SARSA など）が提案され，

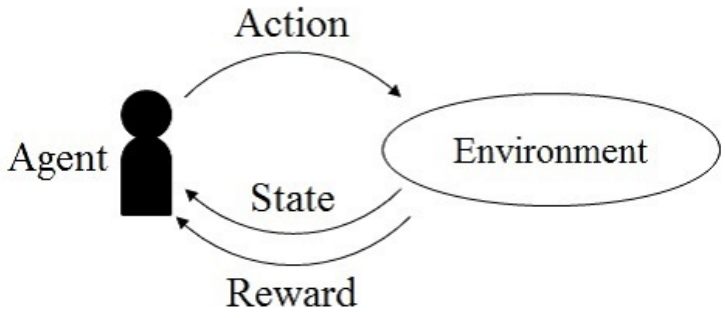


図 2.1 強化学習におけるエージェントと環境のインタラクション

どの手法も一長一短であり全ての問題に対して有効な強化学習器は存在しない．本研究では，他の研究で多く利用されている Q 学習を採用する．

2.2.1 Q 学習

一般的な強化学習アルゴリズムとして Q 学習がある [49]．Q 学習は Q テーブルと呼ばれる Look-up table をもち，その Q テーブルに環境状態に対する行動価値を書き込み方策として構成する．Q テーブルは次式の様に更新される．

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma V(s') - Q(s,a)\} \tag{2.1}$$

$$V(s) = \max_{a \in A} Q(s,a) \tag{2.2}$$

ここで，s はエージェントが観測した環境の現在状態，s' は環境の次状態を示し，$s, s' \in S$ である．a はエージェントが実行可能な行動であり，$a \in A$ である．$Q(s,a)$ は，環境状態 s に対する行動 a の対の集合，すなわち方策を意味する．$\alpha(0 < \alpha \leq 1)$ は学習率であり，$\gamma(0 \leq \gamma \leq 1)$ は割引率，r は報酬である．$V(s)$ は，ある環境状態 s における最大の価値を持つ行動を得る行動価値関数である．また，本研究において，強化学習器で学習した Q テーブル，すなわち方策を知識と呼ぶ．

2.2.2 ボルツマン選択

前節にて説明した Q 学習のアルゴリズムは，通常，実装するために行動選択関数も実装する．これは，Q 学習にて学習した行動価値から，最適な行動を選択するアルゴリズムである．行動選択関数も greedy 方策や ϵ-greedy 方策など，様々なアルゴリズムが存在するが，本研究ではルーレット選択の一種であるボルツマン分布を用いた行動選択関数を用いる．ボルツマン選択は次式のように定義される．

$$p(a|s) = \frac{\exp\left(\frac{Q(s,a)}{T}\right)}{\sum_{b \in A} \exp\left(\frac{Q(s,b)}{T}\right)} \tag{2.3}$$

ここで，a, b は行動であり，$a, b \in A$ である．s は前述の通り環境状態であり，T は行動選択のランダム性を決定する温度定数である．ボルツマン選択は，ある環境状態 s において，行動価値をランク付けし，それを確率的に選択する．各ランクに対しての選択確率を調整するパラメータが T である．

2.2.3　マルコフ決定過程

Q 学習は，マルコフ決定過程 (Markov decision process: MDP) において，無限回の試行回数と最適な学習パラメータのチューニングが行われれば，最適な解が確率 1 で獲得できることが，Watkins らにより証明された [49]．実環境において MDP を再現できる環境は稀であるが，環境を MDP で近似したり，環境を部分観測マルコフ決定過程 (Partially observable MDP: POMDP) とすることにより，Q 学習の有用性が示されている．実問題では，環境状態やエージェントの行動数は有限であるため，有限 MDP である．以降はとくに断りが無い限り有限 MDP を MDP と呼ぶ．ある環境状態 s と行動 a が与えられたとき，MDP は次式で定式化される [29]．

$$\mathcal{P}^a_{ss'} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{2.4}$$

これは，s と a が与えられたときに，環境が s' に遷移する確率を表す状態遷移確率 $\mathcal{P}^a_{ss'}$ である．このとき，次に得られる報酬の期待値 $\mathcal{R}^a_{ss'}$ は次式で表される．次式の r_{t+1} は，次状態 s_{t+1} の時に得られる報酬である．

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} | s_{t+1} = s', s_t = s, a_t = a\} \tag{2.5}$$

したがって，MDP のタプルは $\langle S, A, P, R \rangle$ である．

環境が MDP であることを前提とすると，強化学習の特徴として環境の次状態は，エージェントの行動のみに依存する．すなわち，環境状態の遷移に影響を及ぼすのはエージェントの行動だけであり，図 2.1 のような環境である．また，MDP ではエージェントは環境状態の完全な観測が可能であることが前提となる．

2.2.4　マルチエージェント強化学習と non-MDP

マルチエージェント強化学習は，強化学習エージェントが複数台同時に環境に存在する様なシステムである．したがって，MDP のタプルが $\langle S, A_1, \cdots, A_n P, R_1, \cdots, R_n \rangle$ と書き換えられる [19]．これは図 2.1 に示した MDP 環境から，図 2.2 のような環境になることを意味する．複数台の強化学習エージェントが同時並列的に行動可能であるとすると，環境の次状態への遷移確率は他エージェントの行動にも依存する．すなわち，エージェントがある行動を行っても，そのエージェントは環境の次状態を予測しにくく，強化が学習アルゴリズムを用いても最適な行動を選択できるとは限らない．

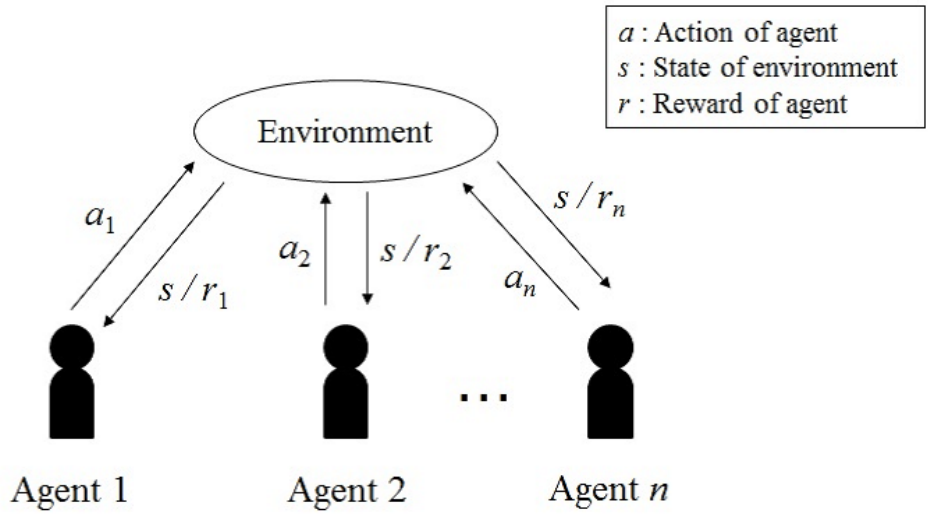


図 2.2 マルチエージェントにおける環境とのインタラクション

したがって，マルチエージェント強化学習では環境を単純に MDP とすることが出来ず，Q 学習では最適解の獲得を保証されない．本論文では，このような MDP 以外の環境を非マルコフ決定過程 (Non-markov decision process: non-MDP) と呼ぶ [26]

実問題を考えると，強化学習ロボットに実装されているセンサの測定距離や範囲は有限である．すなわち，観測可能な環境状態が部分的となる．このような場合，シングルエージェントであっても環境を MDP とすることが出来ず，POMDP となる．POMDP も non-MDP に含まれる．

2.2.5 マルチエージェントによる利点

MARL やマルチロボット強化学習では，環境が non-MDP であり強化学習アルゴリズムを用いても最適な解を学習できるとは限らない．しかし，各エージェントが行動を学習するため，Tan [24] や Matarić [25] の代表的な成果から見てもわかるとおり，協調行動を学習可能である．一般的にマルチエージェントロボットシステムにおける協調行動は，その行動アルゴリズムや行動モデルだけで研究対象となる分野であり，それらを自動的に獲得できる MARL は大きな強みを持っていると言える．

しかし，荒井らが指摘するように MARL では以下の様な問題がある [23, 50]．

- 次元の呪い
- 同時学習問題
- 不完全知覚問題

- 信頼度割り当て問題

しかし，1.2 節でも記した通りこれまで MARL では様々な研究がなされ，システムの効果が示されてきた．例えば，次元の呪いは獲得知識の関数近似により回避可能である．また，non-MDP に由来する同時学習問題は，Matarić [25] などの研究結果からわかるとおり問題にならない場合も多い．したがって，完全な解決が困難な，荒井らが指摘する問題を解決するより，現在示されている優位性を高めることが良いと考える．

2.3　強化学習における転移学習

本節では，本研究の基礎となる手法の 1 つである転移学習についても概説を行う．強化学習エージェントが知識を再利用する手法として，いくつもの成果が発表されている Taylor らの転移学習を基礎として議論を進める．

2.3.1　転移学習の概念

転移学習という概念は心理学用語として古くから存在し，人間が学習した経験や知識を，他の人間に応用させることである [51]．機械学習の領域では 1990 年代から転移学習というメカニズムに着目し，研究が行われている [52]．さらに，強化学習の研究では 2000 年代後半から，転移学習という名前やメカニズムが定着し始めた．

神嶌は，転移学習は “新規タスクの効果的な仮説を効率的に見つけ出すために，1 つ以上の別のタスクで学習された知識を得て，それを適応する問題．すなわち，ある問題を効果的かつ，効率的に解くために，別の関連した問題のデータや学習結果を再利用するのが転移学習である” と述べている [52]．これは，一番受け入れられやすい表現であると考えられる．強化学習における転移学習は “強化学習エージェントが学習した知識を，類似タスクにおいて他の強化学習エージェントが再利用する” ことであると言える（図 2.3）．

2.3.2　強化学習エージェントにおける転移学習

強化学習における転移学習の研究は，Taylor の転移学習 [40] が一番有名であると考えられる．文献 [40] は，Taylor の博士論文が基となっており，2007 年には，転移学習という名前が使われていないものの，現在の Tyalor の転移学習に近い手法が発表されている [53]．強化学習エージェント間における知識の再利用という観点では，2007 年以前にも提案されているが，本論文では議論が活発である Taylor の転移学習手法を前提とする．

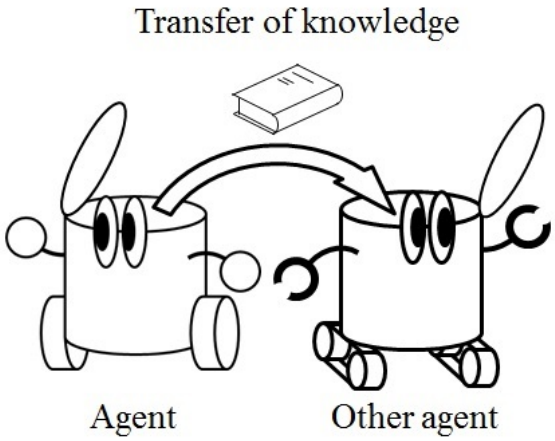


図 2.3 知識の転移のイメージ

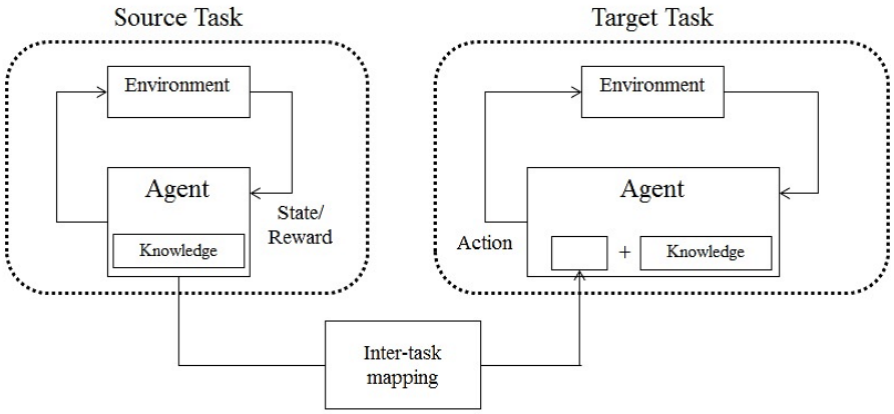


図 2.4 転移学習

2.3.2.1　Taylor の転移学習

Taylor の転移学習（以降，転移学習）は，図 2.4 のようなフレームワークにより実現される．まず，Source task において強化学習エージェントが環境とインタラクションを行い，知識の獲得を行う．次に，Source task で獲得された知識は，Inter-task mapping という処理を通して Target task のエージェントに転移される．Target task の強化学習エージェントは，転移された知識を再利用しながら行動を行い，環境とインタラクションして Target task での強化学習を行う．

例えば Source task のエージェントの観測可能な環境状態集合は $S_{source} = \{s_1^s, s_2^s, s_3^s\}$，行動の集合 $A_{source} = \{a_1^s, a_2^s, a_3^s, a_4^s\}$ とし，Target task のエージェントはそれぞれ $S_{target} = \{s_1^t, s_2^t, s_3^t, s_4^t\}$，$A_{target} = \{a_1^t, a_2^t, a_3^t\}$ とすると，図 2.5 のように各集合の元を対応付ける

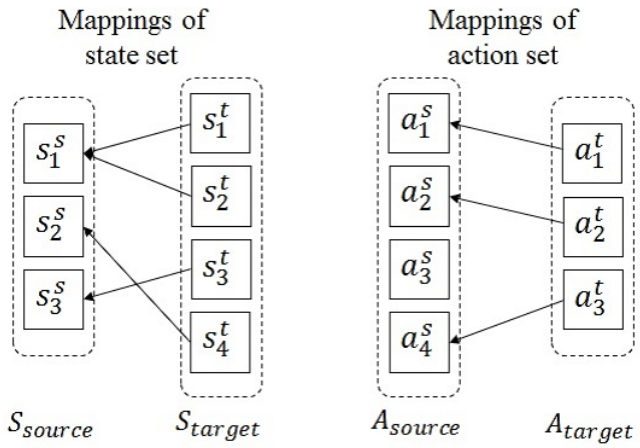


図 2.5 Inter-task mapping の例

処理を行う．言い換えると，Inter-task mapping は Target task の強化学習エージェントが，Source task で獲得された知識をどのように自分が利用するか読み替える作業であるといえる．Inter-task mapping は次式の様に定義される．

$$\begin{aligned} \chi_S(s) &: S_{target} \rightarrow S_{source} \\ \chi_A(a) &: A_{target} \rightarrow A_{source} \end{aligned} \tag{2.6}$$

Target task のエージェントが，Inter-task mapping を用いて Q 学習を行う場合，学習時や行動選択時に次式を用いる．

$$Q^j(s,a) = Q^t(s,a) + Q^s(\chi_S(s), \chi_A(a)) \tag{2.7}$$

このように，Inter-task mapping により再利用可能となった Source task の知識 $Q^s(\chi_S(s), \chi_A(a))$ と，Target task の知識 $Q^t(s,a)$ を結合した，$Q^j(s,a)$ を用いて，学習や行動選択を行う．

上記手法は，文献 [40] において Mountain car problem (2D and 3D) や，ロボットサッカーの様な Keep-away task などで転移学習の有用性を示している．また，知識を転移する際，知識データは膨大な Look-up table となるため，関数近似を用いた転移手法も提案し，CMAC や ANN など知識の関数近似手法を用いた転移学習を評価し，有用性を示した．すなわち，図 2.4 に示した転移学習の方法だけでなく，知識の関数近似を図 2.6 のように利用することも可能である．

さらに，Barrett らはシミュレーションにより獲得した知識を実機ロボットに転移し，転移学習の効果検証を行っている [54]．静的な環境であるため，強化学習ロボットは複

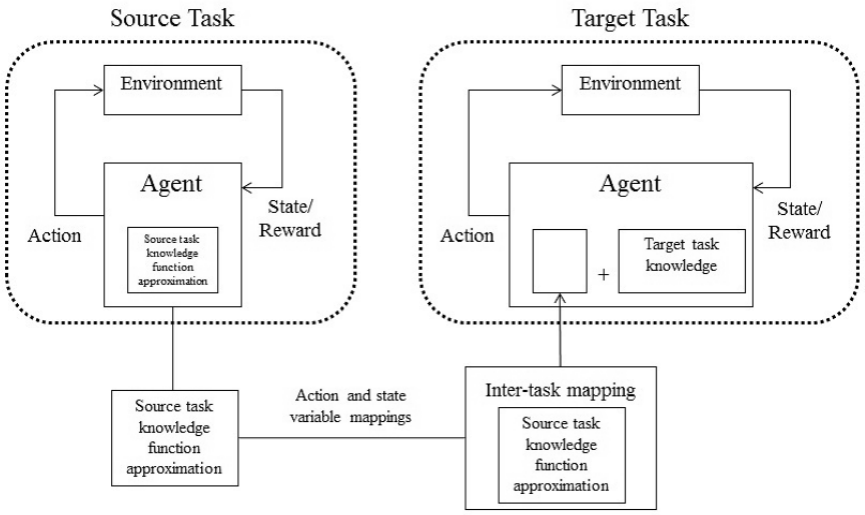


図 2.6 関数近似を用いた Inter-task mapping の概念図

雑なタスクを実行していないが，知識の転移を実機ロボットにて実証している点で有用な知見を得ていると言える．

高野は Taylor の転移学習手法をベースに禁止行動を用いた転移学習手法を提案している [55]．高野の提案手法は，転移学習を行う環境を異遷移同目的タスクと同遷移異目的タスクに分類し，報酬から推測される禁止行動を用いた知識の選択手法を Target task における学習の高速化を目的に，予め複数用意された転移可能知識を，Target task の強化学習エージェントが環境に応じて選択する．タスクにより転移する知識を使い分ける手法などの検討もしており，転移学習以外の機能と統合することで，未知環境に対する確実なタスク遂行を可能にしている．

上記の研究は，基礎的な転移学習をフレームワークとして構成し，応用へ意欲的な研究である．彼らの得た知見は有益であるが，ホモジーニアスなエージェントが前提となっている．

他にも，学習の高速化を行った Celiberto ら [56] や，ヘテロジーニアスなマニピュレータ間の転移学習を行った Lakshmanan ら [57]，Bocsi ら [58] などの研究があり，議論が活発である．

2.3.2.2　転移学習の分類

Taylor の転移学習以外も含む転移学習は，ドメインの差とラベルの有無で手法が分類できる [52]．ドメインとは，強化学習エージェントやロボットに与えられたタスク，行動

表 2.1 ドメインとラベルによる転移学習の分類

		Target domain label	
		Available	No labeled
Source domain label	Abailable	(1) $S+T+$	(2) $S+T-$
	No labeled	(3) $S-T+$	(4) $S-T-$

集合，環境集合を意味する．論文によっては MDP のタプルで表されている [55] ラベルは，分類問題における訓練事例 $\{\mathbf{x}_i, y_i\}$ を例にすると，あるデータ $\mathbf{x}_i$（入力）に対する出力 y_i のことであり，分類結果である．強化学習で言い換えると，入力である s に対する a である．

神嶌 [52] や Pan ら [59] の文献では，ドメインとラベルによって転移学習を表 2.1 の様に分類している．表 2.1 中において，S は Source domain，T は Target domain，$+$ はラベル有り，$-$ はラベル無しを意味する．Source domain や Target domain はそれぞれ，Taylor らの転移学習のにおける Source task と Target task である（強化学習エージェントやロボットも含む）．さらに表中の各数字は，(1) 帰納転移学習 (Inductive transfer learning)，(2) トランスダクティブ転移学習 (Transductive transfer learning) (3) 自己教示学習 (Self-taught learning) (4) 教師無し学習 (Unsupervised transfer learning) である．

本研究における転移学習では，Source task にて獲得した知識において，ラベルが付加されており，また，Target task のラベルは未知であると言える．したがって，本研究で議論する転移学習はトランスダクティブ転移学習である．

2.3.3　マルチエージェント転移学習

前節では，Taylor らの転移学習における基礎的項目を概説した．本節では，Taylor らの転移学習をベースにした近年の応用研究を述べる．

ここで，本章の用語を定義する．シングルエージェントの転移学習とは，Source task や Target task の各タスクで 1 台のみエージェントが存在することとする．マルチエージェントは，Source task と Target task のどちらか，もしくは双方に複数台のエージェントが存在することとする．さらに，ホモジーニアスな転移学習は，Souce task と Target task に存在する全てのエージェントが同一の構成や機能であることを意味する．ヘテロジーニアスは Source task や Target task のいずれか，もしくは双方に構造や機能の異なるエージェントが少なくとも 1 台存在することとする．したがって，前節まで概説した転移学習

はホモジーニアスなシングルエージェント転移学習である．

2.3.3.1　同一マルチエージェント環境における転移学習

Boutsioukis らは，シングルエージェントで研究されてきた転移学習を，マルチエージェント環境で検証した [41]．この研究では，マルチエージェントにおける転移学習実現のために，Bias transfer (BITER) という手法を提案してる．BITER では，複数台エージェント間の ITM を個別に定義するのではなく，複数台のエージェントの行動集合を結合して ITM を行うことで，マルチエージェントにおける ITM を実現している．しかし，BITER ではホモジーニアスな MARL における転移学習を想定している．

Taylor らは，ホモジーニアスなマルチエージェント環境における転移学習の高速化を目的に，複数のタスクで同時並列的に転移学習を行う Parallel transfer learning (PTL) を提案している [42,60]．PTL は，転移学習における Source task と Target task を同時に実行し，常に知識の転移を行うことで，転移学習の高速化をおこなっている．環境としてはマルチエージェント環境であると言える．Taylor らはスマートグリッドの計算機実験により PTL を評価しているが，Taylor らの研究においてもホモジーニアスが前提となっている．

ホモジーニアスマルチエージェントにおける転移学習は他にも文献 [61,62] なども存在し，議論が活発である．

2.3.3.2　異種マルチエージェント環境における転移学習

Taylor らの転移学習は，図 2.5 に示した Inter-task mapping から見て取れるが，ヘテロジーニアスなエージェント間における転移が原理的に可能である．事実，Barrett らは強化学習エージェントから強化学習ロボットへの転移学習が行われている [54]．また，明示的にヘテロジーニアスとして議論を行っている，Lakshmanan ら [57]，Bocsi ら [58] などの研究もある．すなわち，ホモジーニアスなマルチエージェントにおける転移学習（文献 [41,42,60] など）はヘテロジーニアスへ応用が可能である．

しかし，ヘテロジーニアスなマルチエージェント環境における転移学習の検討は，議論がほとんどなされていないのが現状である．

2.4 結言

本章では，強化学習や Q 学習，転移学習などの本研究における基礎的な要素技術について概説した．また，転移学習に関しては近年の研究を概説した．それらの分類を表 2.2 に示す．また，表 2.2 において，ヘテロジーニアスなマルチエージェントにおける転移学習へ応用可能だと考えられる研究は，下線付きで記載する．

表 2.2 に示す通り，ヘテロジーニアスなマルチエージェントを前提とした転移学習に関しては，議論がほとんどなされていない．しかし，このヘテロジーニアスなマルチエージェントにおける転移学習は，本論文における議論点の 1 つである．次章に，知識共創フレームワークの要素である，ヘテロジーニアスを前提としたマルチエージェントにおける転移学習手法を述べる．

表 2.2 近年の転移学習研究の分類

	Single agent	Multi agent
Homogeneous (or similer)	Taylor [40] Celiberto et al. [56] 高野ら [55]	Taylor et al. [42, 60] Boutsioukis et al. [41] Fernández et al. [61] Vrancx et al. [62]
Heterogeneous	Barrett et al. [54] Lakshmanan et al. [57] Bocsi et al. [58]	Taylor et al. [42, 60] Boutsioukis et al. [41]

第3章　Hetero-MARLにおける転移学習法

3.1　諸言

本章では，ヘテロジーニアスなマルチエージェントを前提とした転移学習手法に関して議論を行う．表 1.1 に示した通り，本章は 1.4 節に提示した 3 つの議論点の内，以下の 2 つにアプローチを行う議論である．

- ヘテロジーニティに関する議論
- 不確定エージェントにおける転移学習

これまでの転移学習では，ヘテロジーニアスなシングルエージェント間の転移学習 [54, 57,58] や，ホモジーニアス（類似を含む）なマルチエージェント間の転移学習 [41,42,60–62] が検討されている．ヘテロジーニアスかつマルチエージェントにおける転移学習はほとんど議論されていない．これは，極端に構造や特徴の異なるエージェント間における転移学習は，効果的でないと考えられるからである．例えば，車輪型ロボットで学習した知識が，人型ロボットで活用できるとは直感的には考えにくい．また，ヘテロジーニアスなエージェント間の転移学習を阻害する要因として，Inter-task mapping の設計が困難であるという点が考えられる．例えば，車輪型ロボットで任意の速度で車輪回転させるという行動は，人型ロボットの右足を動かす，というような行動にマッピングできない．したがって，以下の課題を設定する．

課題 1：「ヘテロジーニアスを前提とした Inter-task mapping の検討が必要」

また，マルチエージェントにおける転移学習において，エージェントが多種多様であればあるほど Inter-task mapping の量が増大する．研究レベルでの検討であれば，全てのエージェント間における Inter-task mapping を手動で設計すればよい．しかし，運用・実用段階においては，作業量増大が深刻な課題となる．

さらに，エージェントの故障や修理，それに伴うマイナーチェンジ，新発売のエージェントの導入，老朽化によるエージェントのリプレースなど，エージェントのヘテロジー

ニティは，一定でないことが容易に想定でき，不確定である．異なるエージェントが増えることによって，ここでも，Inter-task ampping 作業量増大の問題が発生すると考えられる．さらに，以下の課題も設定する．

課題 2 :「ヘテロマルチエージェント間における Inter-task mapping の設計作業量増大」

本章では，オントロジを用いて個別のエージェント間で定義されてきた Inter-task mapping を統合，記述を支援する手法である階層的転移学習 (Hierarchical transfer learning:HTL) を提案する．これにより，ヘテロジーニアスなマルチエージェントの転移学習を実現する．

3.2　提案手法の準備

本節では，まずエージェントのヘテロジーニティを定義する．次に，Q 学習にて獲得する知識の形式に関して概説・定義する．以下から述べる知識の形式は，本論文の強化学習ロボットやエージェントの全てに共通することである．

3.2.1　エージェントのヘテロジーニティ

本論文では，ヘテロジーニアスなマルチエージェント間における転移学習を課題としているが，ここでは，どのようなエージェントのヘテロジーニティが考えられるのか整理する．

まずシンプルな例として，エージェントを系と捉えると環境とインタラクションを行うインタフェースは，センサとアクチュエータである．ロボットで例えるならば，センサはカメラや距離センサであり，アクチュエータは駆動台車やアーム，エンドイフェクタとなる．さらに，センサは系の入力に相当する役割があり，アクチュエータは出力に相当する．このシンプルなエージェントのモデルを図 3.1 に示す．

エージェントにおいて，考慮すべきヘテロジーニティはインタフェースである．エージェントの内部モデルは，インタフェースにより隠蔽・抽象化されているためであり，環境との直接的な関係がないからである．すなわち，ヘテロジーニティはセンサやアクチュエータの違いと定義できる．これらの違いの分類を，表 3.1 に示す．

本論文におけるヘテロジーニアスは，表 3.1 中の (4) の一番ヘテロジーニアスな条件を対象とする．(2) や (3)，(4) は，それぞれヘテロジーニティの度合いが異なり，特に (4) はヘテロジーニティが高くなる．そのため，基礎的な実験では (3) などのヘテロジーニティが低いシチュエーションも明示的に用いる．

表 3.1 エージェントのヘテロジーニティ

		Executable actions	
		Homo	Hetero
Observable states	Homo	(1) Homogeneous agent	(2) Heterogeneous actuator
	Hetero	(3) Heterogeneous sensor	(4) Heterogeneous agent

3.2.2 知識の形式

通常，強化学習により獲得される知識は，2.2.1 節で述べたとおり，Q テーブルと呼ばれる Look-up table に記述される．また形式的には，エージェントが観測した環境状態 s を入力として，それに対応する行動 a を検索する形で利用する．すなわち，一種の関数であると言える（図 3.2）．

この Q テーブルの操作は，厳密には，エージェントの行動選択時にはボルツマン選択（式 (2.3)）のような行動選択関数に制御され，学習時には Q 学習の価値更新式（式 (2.1)）によって支配的に制御される．

Q テーブルは Look-up table であるため，入力である環境状態 s の数だけ行を作成する必要がある．すなわち，エージェントが存在する環境の，取り得る環境状態数が多ければ多いほど，Q テーブルは膨大な空間となる．さらに，例えば環境状態集合 $S = \{s_1, s_2, s_3\}$ であった場合，取り得る環境状態数は $s_1 \times s_2 \times s_3$ である．したがって，各元の状態数が多いほど，指数関数的に S の取り得る全環境状態数が増加する．これが次元の呪いである．小規模な環境にいける学習であれば，環境状態数も少なく近年のメモリ大容量化も助け，Q テーブルの全テーブルを計算機内のメモリに保持することができる．しかし，それでは複雑な環境認識が出来ない事を意味し，また計算機リソースも無限ではないため，Q テーブルを関数近似により圧縮する技術が，以前から研究さている [29]．本論文の一部の実験では，Q テーブルの圧縮に関数近似を採用する．次節に Q テーブルの関数近似

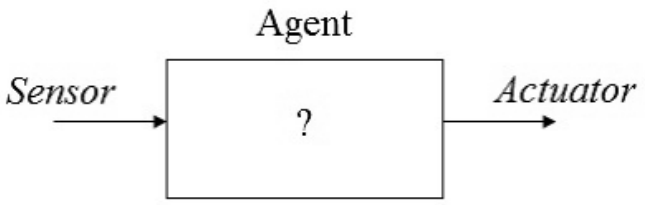


図 3.1 エージェントのシンプルなモデル化

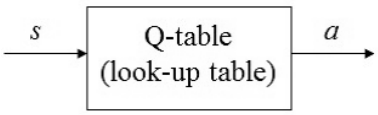


図 3.2 Q テーブル

手法を概説する．

3.2.3　知識の関数近似

強化学習により獲得した知識の関数近似法は，これまでいくつも提案されている．例えば，Artificial neural network (ANN) や Cerebellar model arithmetic computer (CMAC)，Radial basis function (RBF) などがある．これらの関数近似法を用いて，強化学習や転移学習で成果を得ている研究も多い [27, 29, 40, 63]．

強化学習アルゴリズムと同様に，それらの関数近似法は一長一短であり，ケースバイケースで，適宜用いることが重要である．本論文では，関数近似法に ANN を採用することとする．ANN は，プログラム言語におけるライブラリやサンプルなど，情報が充実しており実装が容易である．

図 3.3 に，関数近似の例を示す．この例では，追跡問題のエージェントの Q テーブルであり，学習者であるハンター 2 台，非学習者である獲物 1 台，グリッドワールドのサイズを 7 × 7 としている．ハンターの観測可能な環境状態集合 $S = \{$ 自己位置, 仲間ハンターの位置，獲物の位置 $\}$ であり，行動は $A = \{$ 前進, 後進, 右移動, 左移動, 停止 $\}$ が行えるとする．図 3.3 の上部のように構成される Q テーブルは，図 3.3 の下部のように ANN で近似される．ANN の入力層と出力層は S と A の各元の数に合わせたノード数とし，隠れ層は任意の数のノードをセットアップする．また，本論文では誤差逆伝搬法を用いて ANN の学習を行う．この構成のように関数近似を行えば，ノードの接続関係と，ノード間のネットワーク重みの 2 種類のパラメータを記述すだけで，知識を取り扱える．

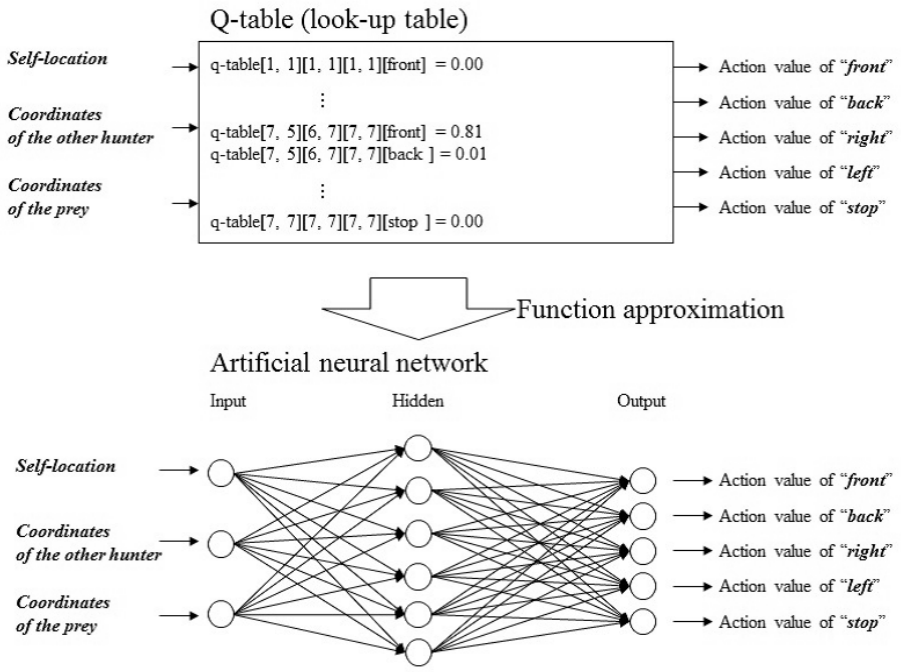


図 3.3 ANN による Q テーブルの関数近似例

3.3　階層的転移学習

ヘテロジーニアスなマルチエージェントの転移学習を実現するために，階層的転移学習を提案する．階層的転移学習は，オントロジにより Inter-task mapping を統合し，ヘテロジーニアスなエージェント間の転移学習を実現すると共に，その設計作業を支援する．

3.3.1　オントロジ

ここでは，オントロジに関する基本的な概念を述べる．オントロジは，哲学で重要な理論であった用語であり “存在論” のことである.「存在」に関する議論は紀元前から行われている，歴史の古い学問領域である [64]．また，近年はこのオントロジの工学的応用が進められており，“概念化の明示的な規約” という，溝口が紹介している Gruber の定義が一般的であると言える [65, 66]．

工学的な応用では，物事の意味や概念を分類，関係性を定義する理論やフレームワークとして研究が行われている．オントロジーを構築するツールとして様々なアプリケーションソフトウェアや，オントロジ用コンピュータ言語が配布されている [67]．例えば，Jena [68] や FIPA [69]，Ontolingua [66] が有名である．特に近年では，人々の知識を共有/再利用する活動にオントロジを活用する検討が行われ [70]，ロボットの情報共有手法として応用を行った研究も行われている [71]．本論文では，特に断りのない限り，工学的応用で用いられているフレームワークやツールのことをオントロジを呼ぶこととする．

単純なオントロジの例を図 3.4 に示す．ロボットの形を分類するオントロジである．オントロジは図 3.4 のように，ツリー階層で表現されることが多い．最上位階層にある Robot は，全体を包括する概念であり，そのオントロジを表現する名前でもある．これは，そのオントロジの中で一番抽象的な概念である．その下位階層に Mobile robot と Humanoid robot という概念が結びついている．さらにその下位階層には Wheel robot と Crawler

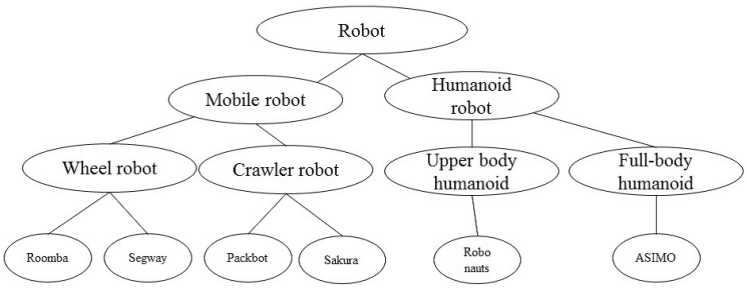
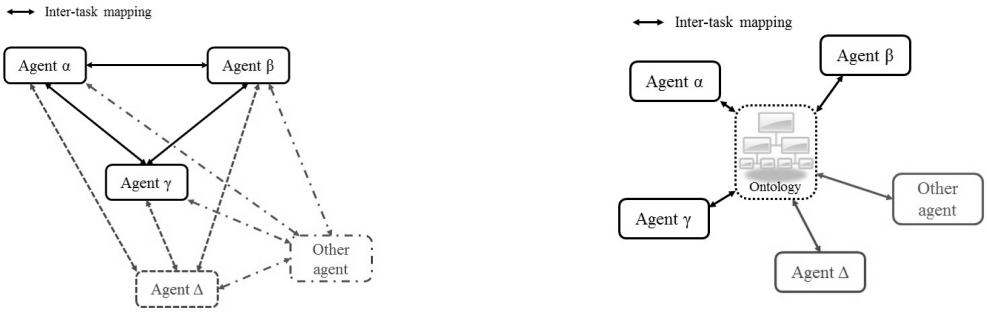


図 3.4 ロボットのオントロジの例



(a) Inter-task mapping　　(b) OITM

図 3.5 OITM による Inter-task mapping の統合

robot，Upper body humanoid，Full-body humanoid という概念が接続されており，これらはクラスと呼ぶ．これは Robot という一番抽象的な概念に対する，具体的な概念であり，下層のクラスになればなるほど概念は具体化される．それぞれの最下層クラスの下には，具体的な概念が接続され，ここではロボットの名前を示している．これをインスタンスと呼ぶ．また，クラス間やクラス–インスタンス間を結ぶ線 “is–a” 関係と呼ぶ．

このオントロジでは，各インスタンスのロボット名が，どの種類の概念（ロボットの形）に分類されるかを示しており，それらのロボットの関係性も明らかにすることが可能である．

オントロジは，図 3.4 のようなマッピング作業を，エキスパートが行い，他のエキスパートと合意を取ることでオントロジの妥当性を確保する．

3.3.2 Inter-task mapping への適用

これまでの Inter-task mapping は，Source task における環境状態集合 S と行動集合 A と，Target task における S と A の写像関係を定義する手法である．さらに，1 対 1 のエージェント同士の写像関係を手動で定義する事が前提であり，多くの種類が存在する転移学習では，各エージェント間で個別に記述する必要がある．本論文では，個別に定義されていた Inter-task mapping を，オントロジで統合して記述する Ontology-base inter-task mapping (OITM) を提案する（図 3.5）．

OITM は，エージェントの S や A の元を，インスタンスとして解釈し，オントロジへマッピングを行う．その例を図 3.6 に示す．ここでは，行動に関するオントロジを示す．各エージェントの行動集合を，エージェント α は A_α とし，エージェント β は A_β，エージェント γ は A_γ とする．また，各エージェントの行動集合の元は以下の通りとする．

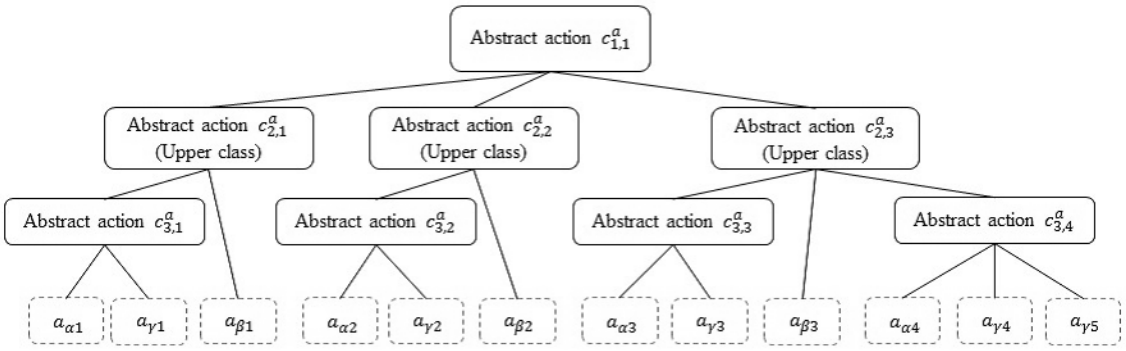


図 3.6 行動オントロジの例

$$
\begin{aligned}
A_\alpha &= \{a_{\alpha 1}, a_{\alpha 2}, a_{\alpha 3}, a_{\alpha 4}\} \\
A_\beta &= \{a_{\beta 1}, a_{\beta 2}, a_{\beta 3}\} \\
A_\gamma &= \{a_{\gamma 1}, a_{\gamma 2}, a_{\gamma 3}, a_{\gamma 4}, a_{\gamma 5}\}
\end{aligned} \tag{3.1}
$$

これらをオントロジの最下層クラスにマッピングする．ここでは，クラスを抽象化行動と呼ぶ．インスタンスである各行動がマッピングされるクラスは，前述の例の通り抽象化されており，エージェント α の開発者や利用者は，この抽象化概念に A_α の元（行動）をマッピングする．また，エージェント β やエージェント γ も同様に，それらの開発者や利用者が，A_β や A_γ のマッピングを行う．すなわち，抽象化された行動の記述と，エージェントの具体的な行動との Inter-task mapping を行うことで，ヘテロジーニアスなエージェント間の Inter-task mapping 作業を支援する．さらに，一度オントロジへマッピングを行えば，そのオントロジにマッピングされている他のエージェントとの Inter-task mapping が記述可能な手法である．

マッピング後，このオントロジをエージェントたちが接続されている Local area network などのインフラで公開する事により，各エージェントはオントロジを参照する事で，全てのエージェント間の Inter-task mapping を参照する事が可能となる．エージェントへの実装を考慮し，既存のフレームワークを用いてオントロジの統一された記述を行えば，オントロジは以下に示す集合や関数の形式でエージェントに実装が可能となる．ここでは，エージェントの行動はオントロジへマッピング済みであることとし，そのオントロジーが記述されたデータは，エージェントが自由にアクセス可能であるとする．

まず，オントロジの各クラスを以下のように記述する．

$$\begin{aligned} C_1^A &= \{c_{1,1}^a\} \\ C_2^A &= \{c_{2,1}^a, c_{2,2}^a, c_{2,3}^a\} \\ C_3^A &= \{c_{3,1}^a, c_{3,2}^a, c_{3,3}^a, c_{3,4}^a\} \end{aligned} \tag{3.2}$$

ここで，C_h^A は各階層におけるオントロジのクラスの集合であり，h は階層番号を意味する．A は行動に関するオントロジを意味する添え字であり，環境状態に関するオントロジの場合は，S となり C_h^S のように記述する．階層が下がる（h が大きくなる）につれて，元 $c_{h,n}^a$ の n が多くなるように見えるが，n は任意の数であり，マッピングするエージェントの種類や，その多さに依存する．必ずしも階層が深くなれば，元が多くなるとは限らないことに注意が必要である．また，C_1^A は，全ての元を包括するクラスとなるため，OITM として特別な機能はないが，オントロジの名前や，オントロジ検索時の基準として利用する．

従来の Inter-task mapping は式 (2.6) で定義されている．OITM も式 (2.6) を基にしたマッピングを，次式の様に定義する．ここでは，環境状態に関するオントロジも定義する．

$$\begin{aligned} \chi_S^O(s) &: S \to C_h^S \\ \chi_A^O(a) &: A \to C_h^A \end{aligned} \tag{3.3}$$

ここで，$\chi_S^O(s)$ と $\chi_A^O(a)$ は OITM を意味する．さらに，各クラスも上位クラスへのマッピングが可能なため，次式を定義する．

$$\begin{aligned} \chi_S^O(c^s) &: C_h^S \to C_{h-1}^S \\ \chi_A^O(c^a) &: C_h^A \to C_{h-1}^A \end{aligned} \tag{3.4}$$

これらの関数を用いて，各エージェントのマッピングをオントロジから検索する関数を次式のように準備する．これにより，オントロジにより記述された ITM をエージェントが利用可能な形として実装する．

$$\begin{aligned} s_k &= \chi_S^O(s,k) \\ a_k &= \chi_A^O(a,k) \end{aligned} \tag{3.5}$$

ここで，k はエージェントの識別子であり，再利用する知識を獲得したエージェントを意

味する．本節の例では α や β，γ である．これにより，オントロジは Inter-task mapping を検索する一種のデータベースとして機能させる．

図 3.6 の $c^{a}_{3,4}$ では，エージェント α の行動 $a_{\alpha 4}$ に対して，エージェント γ の行動 $a_{\gamma 4}$ と $a_{\gamma 5}$ の 2 つの行動が，マッピングされている．したがって，エージェント γ はマッピングされている 2 つの行動のどちらかを選択しなければならない．本論文では，エージェント γ が $a_{\gamma 4}$ と $a_{\gamma 5}$ のどららかを，確率 1/2 で選択する事とする．n 種の行動が同じ抽象化階層にマッピングされた時は，単純に確率 $1/n$ で均等に選択する事とする．

3.3.3　OITM を用いた知識の再利用方法

オントロジにより，各エージェントの s や a のマッピングを検索する関数 $\chi^{O}_{S}(s,k)$ や $\chi^{O}_{A}(a,k)$ を用いて，Target task のエージェントは次式のように知識の再利用を行う．

$$Q^{j}(s,a) = Q^{t}(s,a) + \tau Q^{s}(\chi^{O}_{S}(s,k), \chi^{O}_{A}(a,k)) \tag{3.6}$$

従来の転移学習における知識の再利用式 (2.7) と同様に，ここでは，$Q^{t}(s,a)$ は Target task で学習を行う知識，$Q^{s}(\chi^{O}_{S}(s,k), \chi^{O}_{A}(a,k))$ は再利用する知識，$Q^{j}(s,a)$ は結合知識である．また $\tau \in \mathbb{R}^{+}$ は，本論文では転移率と呼び，知識に記述されている行動価値の度合いを調整するパラメータである．通常，再利用する知識 $Q^{s}(s,a)$ はある程度学習が進んでいる知識であり，十分な行動価値が記録されている．これに対し，Target task にてエージェントが新たに学習する知識 $Q^{t}(s,a)$ は空であると言える．このような状態では，$Q^{s}(s,a)$ と $Q^{t}(s,a)$ の行動価値に大きな差が発生し，$Q^{s}(s,a)$ のみに依存した行動選択行うことが考えられる．すなわち，行動選択時における試行錯誤過程が発生しなくなる．過学習状態が発生するとも言える．これを回避するために，知識の構成要素である行動価値が数値で与えられていることを活用し，行動価値を調整する転移率 τ を，適宜設定しておく．転移率に関する詳細な議論は，次章に示す．提案手法をシステム化した際の概念図を図 3.7 に示す．

図 3.7 において，オントロジはあらかじめサーバに保存され，公開されている．Source task に相当する，知識を獲得したエージェントは，その獲得知識をサーバにアップロードする．次に，Target task のエージェントは，タスクやエージェントのヘテロジーニティを考慮し，適宜再利用する知識を選択しダウンロードする．それと同時に，Target task のエージェントはオントロジもダウンロードしておき，知識の再利用時に活用する．これにより，オントロジと獲得済みの知識をサーバで公開する事で，逐次 Inter-tak mapping の定義作業を必要することなく，エージェントは自由に知識の再利用できる．新たなエー

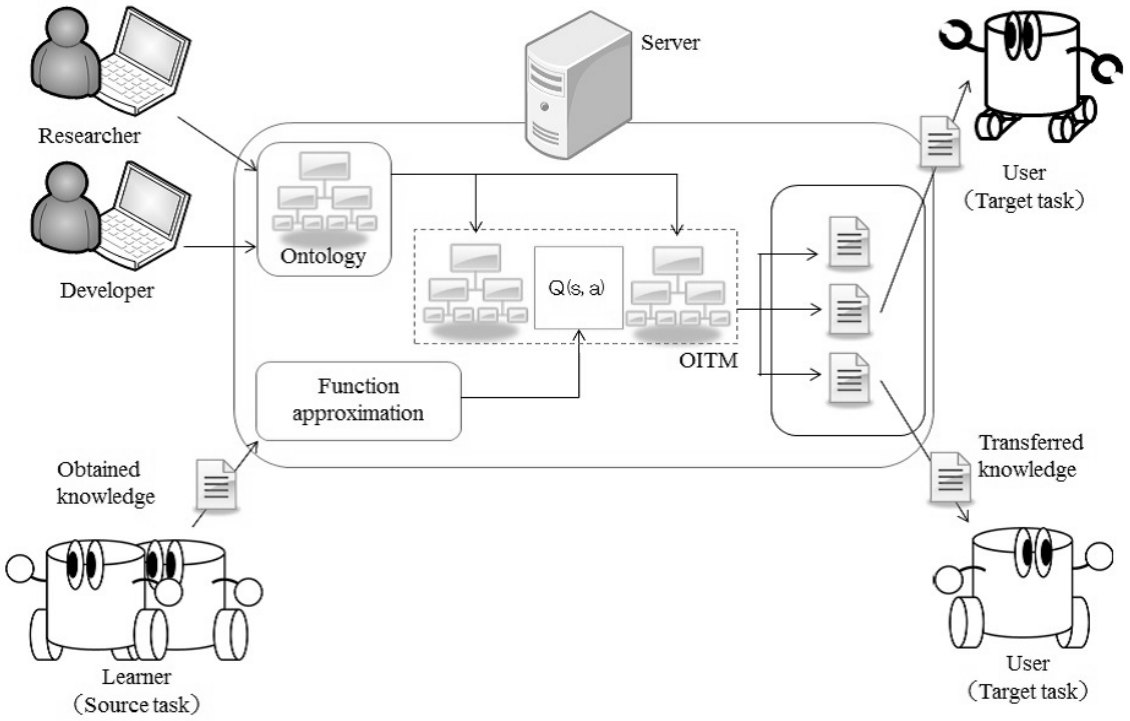


図 3.7 HTL を用いたシステムのイメージ

ジェントがシステムに参加した場合においても，一度オントロジとのマッピングを行えば，他のエージェントの知識を適宜自由に利用する事が可能である．式 (3.6) を実装したエージェントの内部モデルを，図 3.8 に示す．

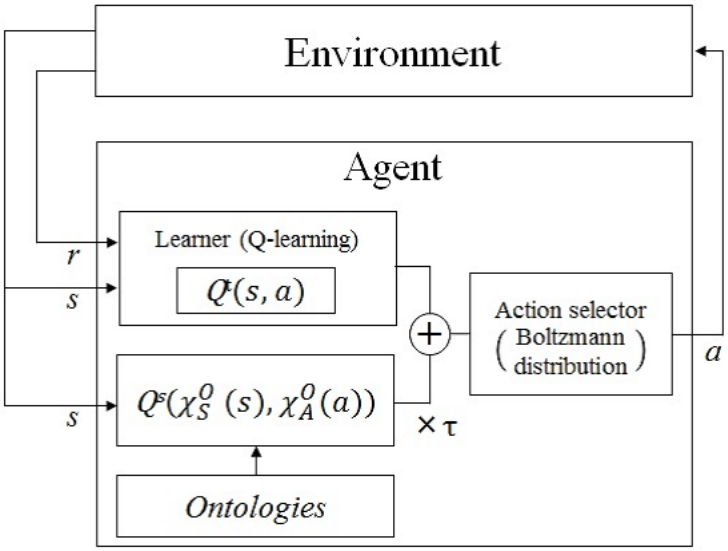


図 3.8 OITM を用いたエージェントの内部モデル

3.4　階層的転移学習の計算機実験

本節では，まず基礎的な実験としてヘテロジーニアスなエージェント間の HTL の効果を，計算機実験にて検証する．本実験の目的は，HTL により知識の再利用が行え，既存の転移学習と同様の効果が得られることを確認することである．言い換えると，Target task において転移無しエージェント（通常の強化学習）より，HTL を用いたエージェントが良いパフォーマンスを示すことを確認する．これは，通常の強化学習と比較して，知識の再利用を行うことで，学習初期からエージェントが高いパフォーマンスを示す，転移学習の効果を確認する事を意味する．本実験では評価関数として，以下から記す追跡問題を採用する．

計算機実験を行うシミュレータ・プラットフォームとしては，付録 A に記すマルチエージェントシミュレータを使用し，数値シミュレーションを行う．シミュレータの詳細は，付録 A にゆずることとする．

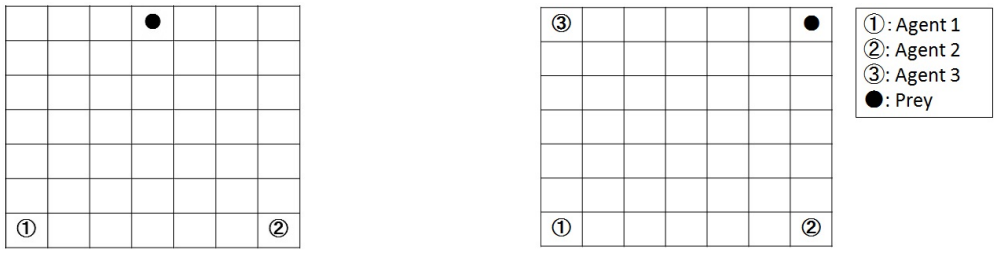
3.4.1　問題設定

追跡問題は，グリッドワールド内をハンターエージェント（以下ハンター）が移動し，獲物エージェント（以下獲物）を捕獲するゲームである．ハンターが獲物を捕獲するまでの行動回数（ステップ数）により，パフォーマンスを評価し，本稿では，Gasser [72] や Tan [24]，Arai らの研究 [21, 22, 50] などを参考に環境設定を行った．

3.4.1.1　環境設定

本実験では，環境としては 7 × 7 のグリッドワールドを用いる．グリッドワールドの周囲四辺を飛び越える移動はできない．エージェントはハンターを 3 台，獲物を 1 台用いる．

本実験では，Source task と Target task で追跡問題の設定が異なる．Source task ではハンターを 2 台，獲物を 1 台用いる（図 3.9(a)）．Target task においては，ハンターを 3 台，獲物を 1 台用いる（図 3.9(b)）．これは，自エージェントの観測可能な他エージェント数が異なる．Source task のハンターと Target task のハンターで観測可能な環境状態がヘテロジーニアスとなる．後述するが，各エージェントの行動もヘテロジーニアスなため，前節の実験よりさらにヘテロジーニアスな MARL にて追跡問題を行うことが，本実験の狙いである．



(a) Source task における初期位置　　(b) Target task における初期位置

図 3.9 追跡問題のエージェント初期位置

(a) Source task (2vs.1 タスク)　　(b) Target task (3vs.1 タスク)

図 3.10 獲物の捕獲条件例

各エージェントの初期座標は図 3.9 に示す配置である．獲物の捕獲条件は，Source task は図 3.10(a) であり，Target task は図 3.10(b) である．全てのエージェントが，獲物と十時方向に隣り合う座標に移動した時とする．各エージェントの行動順序はハンター 1→ ハンター 2→ ハンター 3→ 獲物の順とする．捕獲状態に到達したら各エージェントは初期位置にリセットされる．

3.4.1.2　エージェント設定とヘテロジーニティ

本実験では，ヘテロジーニアスなエージェントを再現するために，図 3.11(a)–(c) のような 3 種のエージェントをハンターとして用い，図 3.11(d) を獲物として追跡問題を行う．これらのエージェントは，移動特性が異なり Source task と Target task で観測可能なエージェント数が異なる．これはエージェント間で行動集合が異なり Source task-Target task 間では状態集合が異なる．

まず，ハンターの行動集合に関して述べる．各ハンターは次式に示す行動集合を有している．

$$A_{hunter1} = \{forward,\ backward,\ right,\ left,\ stop\} \tag{3.7}$$

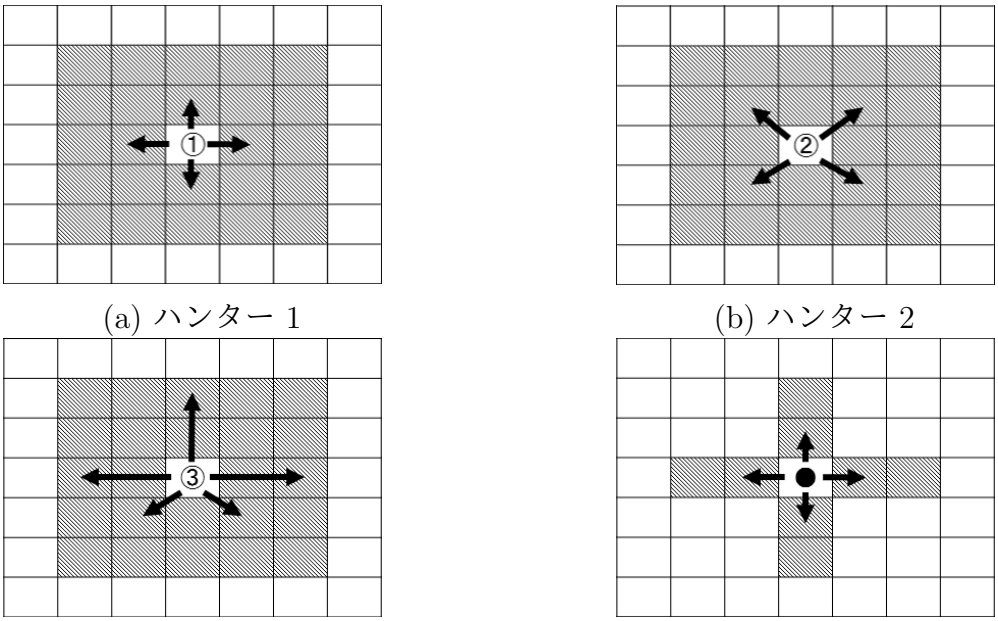
$$A_{hunter2} = \{forward\ right,\ backward\ right,\ backward\ left,\ forward\ left,\ stop\} \tag{3.8}$$

$$A_{hunter3} = \{long\ forward,\ long\ right,\ backward\ right,\ backward\ left,\ long\ left,\ stop\} \tag{3.9}$$

ここで，ハンター 1 からハンター 3 はそれぞれ $A_{hunter1}$ から $A_{hunter3}$ の行動集合に対応している．図 3.11 の上方向矢印は式 (3.7) の forward と対応しており，それを基準に右方向矢印は right など対応している．本実験で用いるハンターは，図 3.11(a)〜(c) に示すグレーにマスクされた範囲が視覚範囲である．全てのハンターは同様の視覚範囲を有する．

状態集合において，Source task と比較して Target task ではハンターの台数が増加しているため，観測可能な環境状態の変数が異なる．Source task の環境状態集合を $S_{2vs.1}$ とし，Target task における環境状態集合は $S_{3vs.1}$ とすると，次式の様に定義できる．

$$S_{2vs.1} = \{\ x\text{-}coordinate\ of\ self,\ y\text{-}coordinate\ of\ self,\ x\text{-}coordinate\ of\ the\ second\ hunter,\ y\text{-}coordinate\ of\ the\ second\ hunter,\ x\text{-}coordinate\ of\ prey,\ y\text{-}coordinate\ of\ prey\} \tag{3.10}$$



(a) ハンター 1　(b) ハンター 2

(c) ハンター 3　(d) 獲物

図 3.11 階層的転移学習の計算機実験に用いるエージェント設定

$$
\begin{aligned}
S_{3vs.1} = \{ \quad & x\text{-}coordinate\ of\ self, \\
& y\text{-}coordinate\ of\ self, \\
& x\text{-}coordinate\ of\ a\ second\ hunter, \\
& y\text{-}coordinate\ of\ a\ second\ hunter, \\
& x\text{-}coordinate\ of\ a\ third\ hunter, \\
& y\text{-}coordinate\ of\ a\ third\ hunter, \\
& x\text{-}coordinate\ of\ prey, \\
& y\text{-}coordinate\ of\ prey\}
\end{aligned}
\tag{3.11}
$$

獲物は $A_{hunter1}$ と同様に，次式に示す行動集合を設定する．

$$
A_{prey} = \{forward,\ backward,\ right,\ left,\ stop\} \tag{3.12}
$$

基本的に獲物はランダムに行動選択を行うが，視界内にハンターが存在する場合，ハンターと逆方向に移動する逃避行動を行う．タスクによらず，視界内においては，獲物は全てのハンターが認識可能である．視界内における獲物の観測可能なハンターの数は，Source task が 2 ハンター，Target task では 3 ハンターとなる．

3.4.1.3　実験条件

本実験では，以下に示す 4 種類の実験条件を用いる．

(1)　自己転移 (Self-transfer)
(2)　異行動集合 (Different action space)
(3)　異状態集合 (Different state space)
(4)　ヘテロジーニアス（異行動異状態） (Heterogeneous)

実験条件 (1) は，Source task で獲得した知識が，正しく生成されているかを検証するために，Source task と同条件の追跡問題へ知識の転移を行う，自己転移実験である．言い換えると，自己転移は，生成した知識を自分自身で再利用する実験条件である．

実験条件 (2) は，エージェントの移動特性（方向と距離）のみが異なる MARL 間で HTL を用いる実験条件である．タスクとしては，2 vs. 1 の追跡問題を用い各エージェンとの観測可能な状態（状態空間）は同一とする．

実験条件 (3) は，実験条件 (2) と異なり行動集合は同一のエージェントを用いるが，Source task と Target task で観測可能な状態（状態空間）が異なる設定である．Source task は 2 vs. 1 を採用し，Target task では 3 vs. 1 とする．これにより，Source task で獲得した知識を再利用する場合，Target task のエージェントは状態空間が異なる知識の再利用を行うこととなる．

実験条件 (4) はこれまで述べた実験条件の中で一番ヘテロジーニティが高い実験条件である．設定としては，実験条件 (2) と実験条件 (3) のハイブリッドである．Source task のエージェントと Target task のエージェントでは，行動空間と状態空間の両方が異なる状態である．

各実験における使用ハンターの種類，タスクの種類は表 3.2 に示すとおりである．本実験では，強化学習器に Q 学習を採用する．いずれの実験条件においても，強化学習の学習パラメータは表 3.3 に示すとおりである．

本実験において，階層的転移学習に用いるオントロジは予めハンター内部に組み込んであることとする．Target task のハンターは，自身にプログラムされたオントロジを逐次参照し，転移された他エージェントの知識を再利用する．本実験で用いるオントロジを図 3.12 に示す．実験用オントロジは，ハンター 1 の行動を基に，式 (3.7) をマッピングしている．設計指針として，同様の方向の行動は同一クラスへマッピングする．例えば，forward と forward right などの前方への移動で進行方向が厳密には異なる行動は，Forward などの上階層のクラスへマッピングを行った．

表 3.2 階層的転移学習の計算機実験における実験条件

Experiment	Conditions	Source task		Target task
Self-transfer	Task	2 vs. 1		2 vs. 1
	Hunters	A1 and A2		A1 and A2
	Direction of transfer	A1	→	A1
		A2	→	A2
Different action space	Task	2 vs. 1		2 vs. 1
	Hunters	A1 and A2		A2 and A3
	Direction of transfer	A1	→	A2
		A2	→	A3
Different state space	Task	2 vs. 1		3 vs. 1
	Hunters	A1 and A2		Two A1 and one A2
	Direction of transfer	A1	→	A1
		A2	→	A2
Heterogeneous	Task	2 vs. 1		3 vs. 1
	Hunters	A1 and A2		A1, A2, and A3
	Direction of transfer	A2	→	A1
		A1	→	A2
		A1	→	A3

A1: Agent 1, A2: Agent 2, A3: Agent3.

表 3.3 異行動集合を用いた計算機実験におけるパラメータ設定

Parameter	Value
Learning rate	0.1
Discount rate	0.99
Reward	1
Boltzmann parameter	0.01
Default Q-value	0
Transfer rate	0.5
Number of episode of source task	10000
Number of episode of target task	10000
Number of trial	10

環境状態オントロジは図 3.13 に示す．各エージェントで自己位置は全て同じクラスにマッピングされており，すなわち知識を再利用するエージェントは，観測した自己位置をそのまま再利用する知識の自己位置入力に入力する．ハンターが観測した獲物も同様である．表 3.2 の全ての Source task は 2 vs. 1 タスクである．したがって，Source task で自エージェントの他に協調できるハンターは 1 台のみである．Target task では協調可能なハンターは 2 台であるため，知識を再利用する際は協調可能なハンターのどちらかの座標を知識に入力する．例えば図 3.13 において，Agent 3 が Agent 1 の知識を再利用することを考える．仮に Agent 3 が協調可能な Agent 1 と Agent 2 の両方を観測できたとする．Agent 3 は Agent 1 の座標をオントロジに入力し，再利用知識には Agent 2 の座標として入力される．言い換えれば，Source task で Agent 1 と Agent 2 が協調行動可能な知識を Agent 3 が再利用した場合，Target task では Agent 3 と Agent 1 が協調行動を実行する．

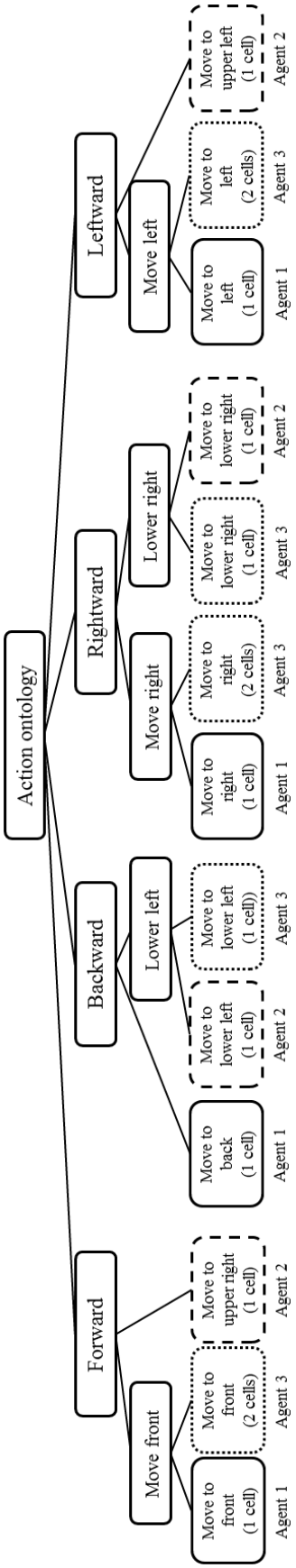


図 3.12 実験に用いる行動オントロジ

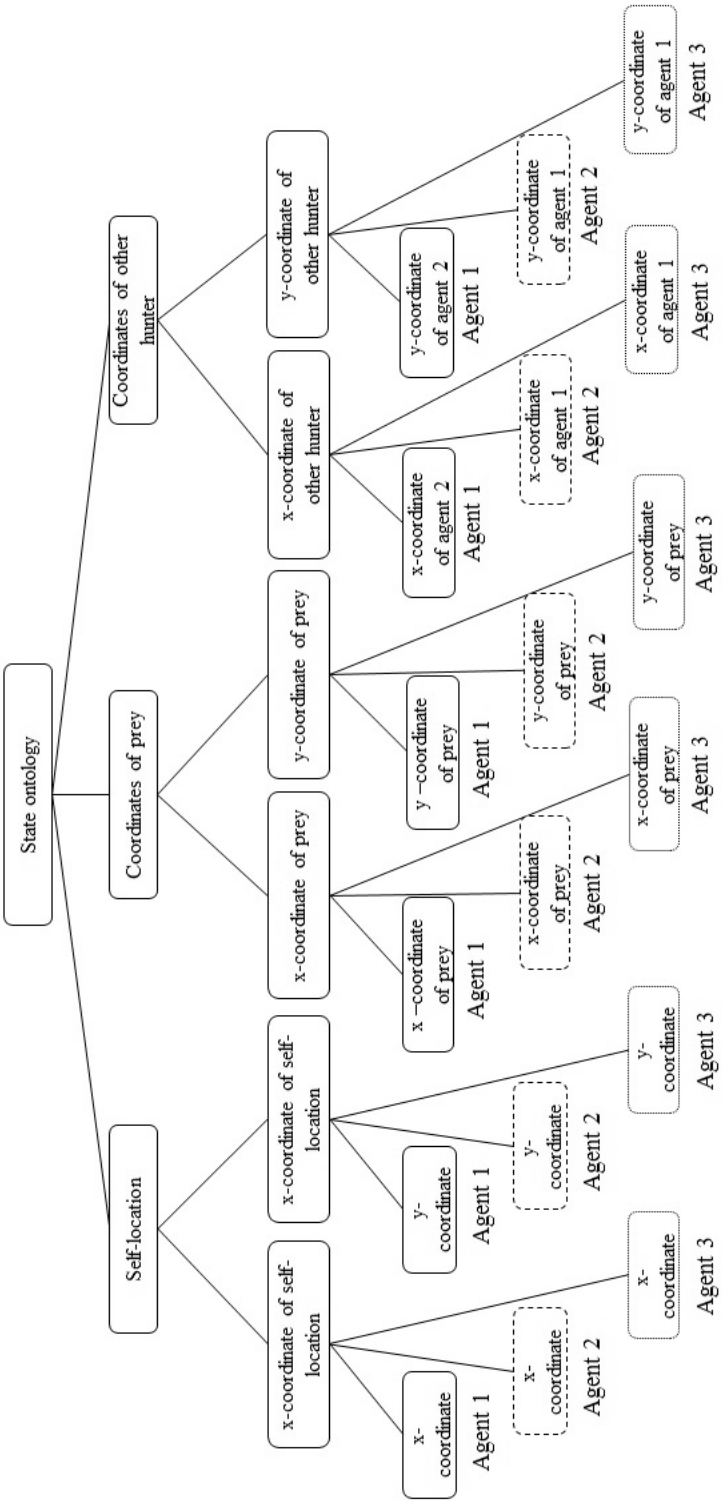


図 3.13 実験に用いる環境状態オントロジ

3.4.1.4　評価項目

本実験では自己転移と HTL の両実験を，学習の結果を表す学習曲線を用いて評価する．学習曲線は，横軸に学習の繰り返し回数であるエピソード，縦軸に目標を達成するまでに要した行動回数をとる曲線である．この学習曲線が，学習の繰り返しとともに低い値をとれば，学習によるエージェントのパフォーマンス向上が表れていることとなる．具体的には，ジャンプスタート (Jump start : JS) と，最終捕獲ステップ数平均の差 (Difference in convergence steps:DCS) を評価する．

JS は初期エピソードにおいて，転移無し学習（通常の強化学習）と比較して，転移有りのエージェントのパフォーマンスがどれだけ高いかを表す定量的な指標である．Target task における転移無しの学習曲線において，あるエピソード i におけるステップ数を $S_i^n t$, 同様に Target task における転移有りの学習曲線におけるステップ数を S_i^t とし，JS を次式で定義する．

$$\mathrm{JS} = \frac{1}{100}\left(\sum_{i=1}^{100} S_i^{nt} - \sum_{i=1}^{100} S_i^t\right) \tag{3.13}$$

また，JS だけでなく，転移無しの学習と対比をとるために，JS の比率を Ratio of JS (RJS) として，次式の様に定義する．

$$\mathrm{RJS} = \sum_{i=1}^{100} S_i^t \Big/ \sum_{i=1}^{100} S_i^{nt} \tag{3.14}$$

DCS は転移無しと転移有りのエージェントが，最終的に同等のパフォーマンス，もしくはことなる異なるパフォーマンスを獲得するか確認するための指標である．JS に良い効果が表れていたとしても DCS が悪い場合，良い転移の結果とは言えないため，DCS による比較を行う．式 (3.13) や式 (3.14) のように，DCS と Ratio of difference in convergent steps (RDCS) を以下に定義する．

$$\mathrm{DCS} = \frac{1}{100}\left(\sum_{i=9901}^{10000} S_i^{nt} - \sum_{i=9901}^{10000} S_i^t\right) \tag{3.15}$$

$$\mathrm{RDCS} = \sum_{i=9901}^{10000} S_i^t \Big/ \sum_{i=9901}^{10000} S_i^{nt} \tag{3.16}$$

学習曲線における JS や DCS のイメージを図 3.14 に示す．

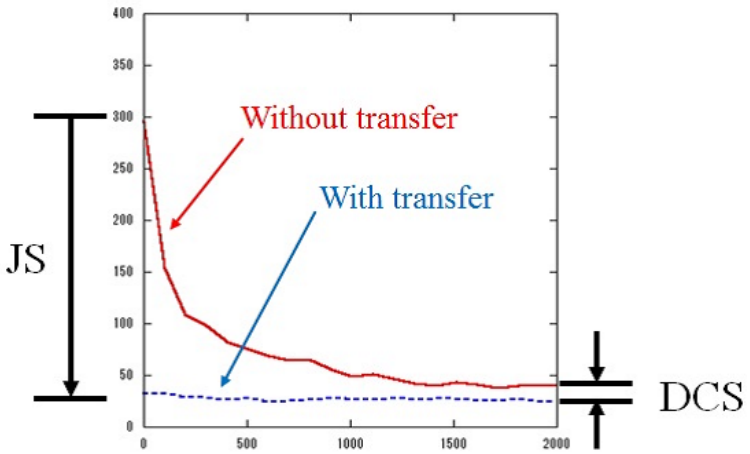


図 3.14 評価指標 JS と DCS のイメージ

3.4.2　実験結果

各実験条件における学習曲線を図 3.15 に示す．それぞれの JS と RJS，DCS，RDCS を表 3.4 に示す．本結果の学習曲線は，膨大なデータの傾向を表すために，付録 B に示すスムージング処理を行いグラフの描画を行っている．

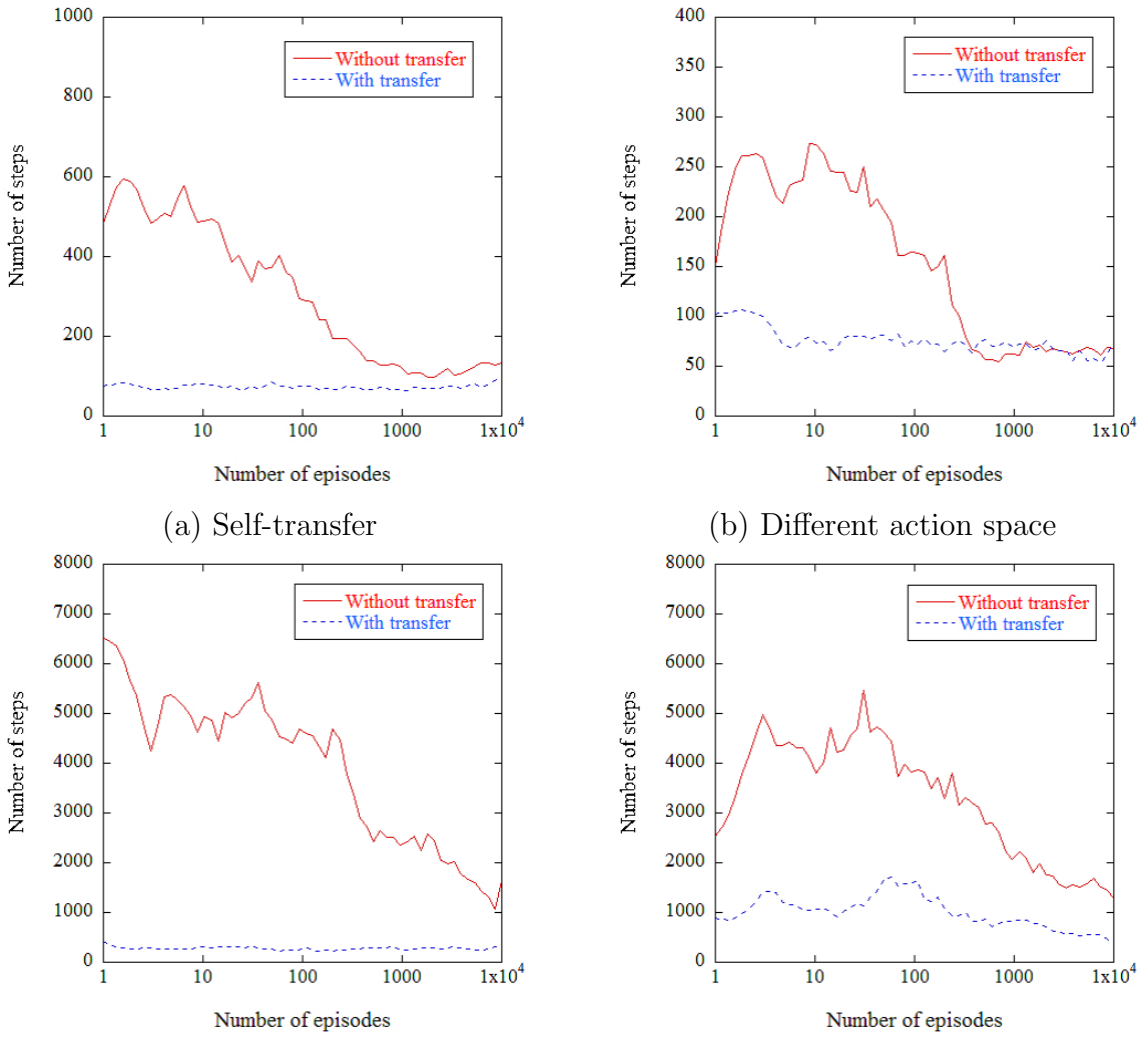
3.4.2.1　自己転移の実験結果

自己転移の実験条件における，転移無しの学習曲線と転移有りの学習曲線を図 3.15(a) に示す．転移無しにおける学習曲線は，動的環境であるため一意のステップ数に収束していない．しかし，明らかなパフォーマンスの向上が見られているため，収束傾向が表れていると言える．

自己転移における JS は 297.16steps であった（表 3.4）．改善率とすると約 80%の JS となり，“Without transfer” と比較して明らかな JS が発現していることが見て取れる．さらに，“With transfer” の学習曲線における最終 100episode 平均は，“Without transfer” の学習曲線の最終 100episode 平均よりも低く，42.84steps の DCS が発現している．RDCS としては 0.64 であり，“Without transfer” と比較して “With transfer” の学習曲線は収束値の改善も発現している．

3.4.2.2　異行動集合の実験結果

異行動集合による HTL の結果である学習曲線を図 3.15(b) に示す．本実験条件においても，明らかな JS が発現している．表 3.4 に示す通り，JS の値としては 108.35steps で



(a) Self-transfer　　(b) Different action space

(c) Different state space　　(d) Heterogeneous

図 3.15 階層的転移学習の計算機実験における実験結果

表 3.4 各実験条件における JS と RJS，DCS，RDCS の比較

Experiment	JS	RJS	DCS	RDCS
Self-transfer	297.16	0.20	42.84	0.64
Different action space	108.35	0.42	-3.64	1.06
Different state space	4433.01	0.06	1095.25	0.19
Heterogeneous	3059.12	0.28	602.67	0.51

ある．これは，“Without transfer” の学習曲線から 58%ものパフォーマンス改善が現れている．本結果より，同一状態集合異行動集合を持つエージェント間における知識の再利用を，HTL により実現できたといえる．さらに，“With transfer” の学習曲線は，その学習曲線の初期エピソードより緩やかであるが値の低下が現れている．この現象は，転移された知識をエージェントが再利用しながら，新たな環境における知識を獲得していると考えられる．

DCS の値として，“With transfer” の学習曲線の収束値は “Without transfer” の学習曲線と比較して 6%高いステップ数となっている．

3.4.2.3　異状態集合の実験結果

本実験条件の学習曲線を図 3.15(c) に示す．本実験条件においても明らかな JS が発現し，その値は 4433.01 steps と大きい．RJS としては 0.06 steps であり，HTL による知識の再利用により大きなパフォーマンス改善を得られている．さらに，RDCS の値としても 0.19 steps であり，大きなパフォーマンス改善が見て取れる．

3.4.2.4　ヘテロジーニアス（異行動異状態）の実験結果

本実験では，状態集合も行動集合も異なる最も転移が難しい実験条件である．結果として，図 3.15(d) に示す通り明らかな JS が発現した．RDCS では 49%ものパフォーマン

ス改善が得られた．この効果は，ヘテロジーニアスな MARL 間における知識の再利用による効果が発現しているといえる．

3.4.3　考察

3.4.3.1　自己転移の考察

自己転移時の実験条件は，知識を転移する側と転移される側のタスクが同様であり，エージェントも同一である．既存研究で示された効果と同様に，最も知識の再利用が行いやすい実験条件である．

実験結果から，明らかな JS も発現し DCS の値も低い．すなわち，提案手法である HTL を用いてホモジーニアスなエージェント間の転移が行えたといえる．この直感的には知識の再利用が行える実験条件において，HTL による転移学習の効果が発現したことで従来の転移学習の効果が得られると共に，Source task において転移する知識が正常に生成でき，ヘテロジーニアスな実験条件で HTL を用いる準備が確認できたこととなる．

3.4.3.2　異行動集合の考察

本実験条件においても，明らかな JS が発現し DCS においては，“Without transfer” と “With transfer” の両学習曲線が同様のステップ数に収束する結果が得られた．

DCS の値として，“With transfer” の学習曲線の収束値は “Without transfer” の学習曲線と比較して高く現れていることは既に述べた．直感的には，この現象は一から環境を学習した時のパフォーマンスより悪くなっていると考えられるが，表 3.4 に示した通り，約 6%違いしかないことがわかる．これは，“With transfer” と “Without transfer” の学習曲線は同等のパフォーマンスにしたと考えられる．したがって，行動集合が異なる MARL 間における知識の再利用が可能であることがいえる．

3.4.3.3　異状態集合の考察

前項の異行動集合と同様に，本実験条件でも明らかな JS が発現し，DCS においては “With transfer” の学習曲線が “Without transfer” を下回るという結果になった．この主な原因は，α や γ，T の学習パラメータのチューニングであると考えられる．強化学習は，環境に対するパラメータに敏感な場合が存在する．本実験条件では，Source task と Target task のエージェントの台数設定が異なる．これにより，“With transfer” の学習曲

線は，通常の強化学習（“Without transfer”）では到達できないパフォーマンスへ収束したと考えられる．

本結果より，異状態集合同一行動集合の関係にあるエージェント間における知識の再利用を，HTL により実現できたといえる．また，これまでの結果より行動集合や状態集合のどちらかが異なる MARL 間の転移学習は，HTL により実現可能であることが示唆される．

3.4.3.4　ヘテロジーニアス（異行動異状態）の考察

本実験条件でも，明らかな JS と DCS が発現したことは既に述べた．これまでの実験結果と比較して “With transfer” の学習曲線は安定していない．この原因はタスクの難易度であると言える．図 3.15(d) における “With transfer” の学習曲線では，JS は発現しているがステップ数が増減し，さらに良いパフォーマンスへの収束が見られる．これは，知識の再利用により学習初期から高いパフォーマンスを発揮したが，新たな環境を学習する試行錯誤過程でパフォーマンスが不安定となり，その後通常の強化学習通り低いステップ数へ収束したと考えられる．本実験条件の Target task は，Source task と比較して行動集合も異なり，状態集合の違いを造るためにタスクが異なる．さらに，知識を再利用するエージェントの関係もヘテロジーニアスである．しかし，この最もヘテロジーニティが高いタスクにおいて，JS が発現し DCS も低いため，HTL による知識の再利用の効果が発現していると考えられる．

最終的な評価として，JS や DCS の発現度合の違いはあるが全ての実験条件において，知識の再利用の効果すなわち転移学習の効果が HTL によりもたらされたと考えられる．

3.5 ANN を用いた HTL の計算機実験

前節の実験では，Source task における獲得知識の関数近似を行わず，階層的転移学習の効果を検証した．本実験では，転移する知識を ANN により関数近似を行い HTL による転移学習を行う．

獲得知識の ANN による関数近似は，3.2.3 項に述べたように獲得知識のデータ量軽減を目的としている．HTL の実世界における運用を考えると，Q テーブルのように膨大な行数の Look-up table をロボット内のメモリにロードすることは現実的でない．そこで，本実験では HTL の運用時は知識の関数近似がなされることを前提に，ANN による知識の近似が HTL にどのような影響を及ぼすか検証することが本実験の目的である．

計算機実験を行うシミュレータ・プラットフォームも前節と同様に，付録 A に記すマルチエージェント強化学習シミュレータを用いる．また，ANN にはマルチエージェント強化学習シミュレータに実装可能な，クロスプラットフォームのライブラリである Fast artificial neural network (FANN) を用いる [73, 74]．

3.5.1 問題設定

本実験では，前節の実験と同様にヘテロジーニアスな MARL における HTL の効果を，追跡問題を用いて評価する．環境設定やエージェントの設定に関して重複する部分も多いため，既に述べられている設定に関しては，引用する形で説明を行う．以下より本実験の設定や条件を述べる．

3.5.1.1 環境設定

本実験においても，前節の実験と同様に環境としては 7 × 7 のグリッドワールドを用いる．グリッドワールドの周囲四辺を飛び越える移動はできない．エージェントはハンターを 2 台もしくは 3 台，獲物を 1 台用いる．ハンターエージェントや獲物の移動特性，視覚範囲は図 3.11 と同様に，3 種類のハンターと 1 種類の獲物を用いる．

各エージェントの初期座標は図 3.16 に示す配置である．Source task と Target task の獲物の捕獲条件は，前節の実験と同様に図 3.10 である．2 台もくしくは 3 台のハンターが同時に獲物と隣り合う座標に移動した時とする．各エージェントの行動順序はハンター 1→ ハンター 2→ ハンター 3→ 獲物の順とする．捕獲状態に到達したら各エージェントは初期位置にリセットされる．

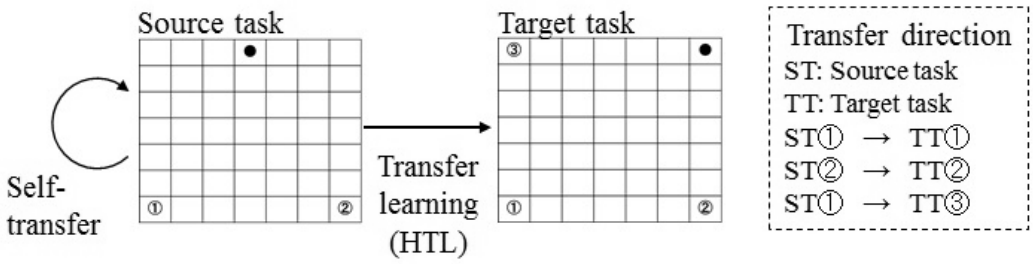


図 3.16 ヘテロジーニアスな MARL における自己転移と HTL

表 3.5 ヘテロジーニアスな MARL を用いた計算機実験における実験条件

	Self-transfer			HTL		
Conditions	Source task		Target task	Source task		Target task
Task	2 vs. 1		2 vs. 1	2 vs. 1		3 vs. 1
Hunters	A1 and A 2		A1 and A2	A1 and A2		A1, A2 and A3
Direction of transfer	A1	→	A1	A1	→	A1
	A2	→	A2	A2	→	A2
				A1	→	A3

A1 : Agent 1, A2 : Agent 2, A3 : Agent 3

3.5.1.2　実験条件

本実験においても，図 3.16 に示す自己転移と HTL の両方の評価を行う．前節の実験と同様に，まず自己転移の実験により，Source task の各エージェントが転移するのに十分な知識を生成できているか評価する．その後，HTL により獲得知識を転移し，HTL の効果を評価する．各実験に用いるエージェントの種類や，知識を転移させるエージェントのペアを表 3.5 に示す．

Source task においてハンター 1 が獲得した知識は，Target task のハンター 1 へ転移される．同様に，Source task のハンター 2 の知識は，Target task のハンター 1 へ転移される．また，Target task の Agent 3 は一番共通な行動の方向を持つ，Agent 1 の知識を再利用することとする．

本実験においては，Q 学習の学習パラメータは表 3.6 に示すとおりである．

ANN のセットアップとして，入力ノード数 = 7，隠れノード数 = 14，出力ノード数 = 5 に設定し，誤差逆伝搬法を用いて知識の近似を行う．獲得知識の ANN による近似の手順として，Source task のエージェントが獲得した知識を一度 Look-up table として計

表 3.6 ヘテロジーニアスな MARL を用いた計算機実験におけるパラメータ設定

Parameter	Value
Learning rate	0.7
Discount rate	0.9
Reward	1
Boltzmann parameter	0.01
Default Q-value	0
Transfer rate (Self-transfer)	1.0
Transfer rate (HTL)	1.0
Number of episode of source task	5000
Number of episode of target task	5000
Number of trial	10
Input node of ANN	7
Hidden node of ANN	14
Output node of ANN	5

算機内のファイルに保存し，それらを ANN の関数近似プログラムで読み込み，関数近似後にノード情報やそれらの接続関係が記された ANN のファイルとして保存する．保存された ANN のファイルを Target task のエージェントが読み込み，ANN として再構築後 ANN のまま知識の再利用する．ANN の関数近似プログラムには，Fast artificial neural network library (FANN) を用いた [73, 74].

本実験では，Source task と Target task で環境状態変数の数が異なる．すなわち，再利用する知識 (ANN) の入力変数の数が Target task では不足する事となる．この場合は，OITM により再利用知識へ入力できる環境状態変数だけ用い，入力できない環境状態変数は無視する事とする．無視する環境状態変数は，Target task で行われる学習により適応する．ここまで説明したパラメータを表 3.6 にまとめる．

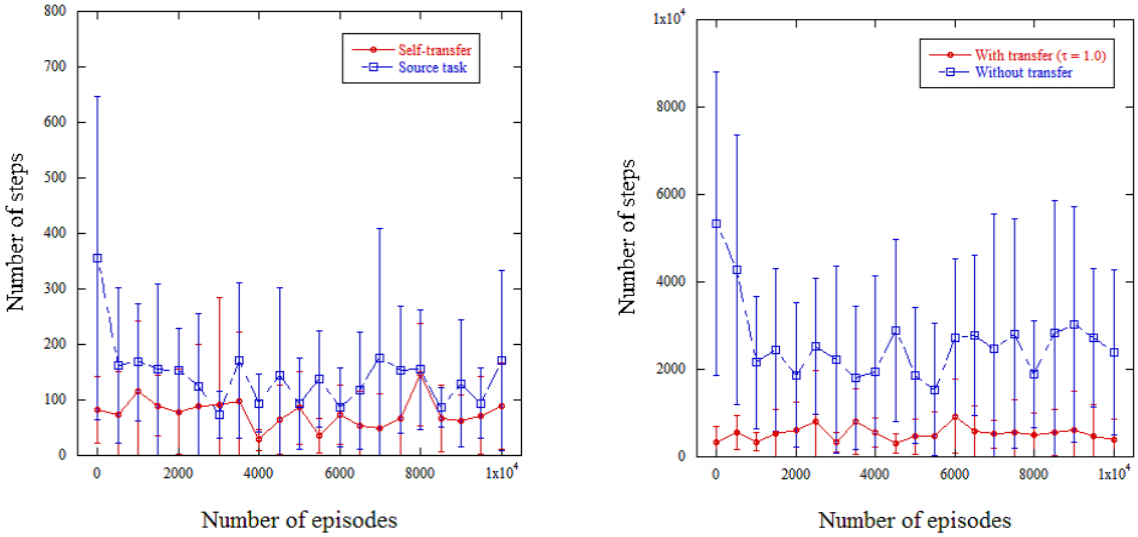
実験に利用するオントロジは，前節の実験と同様に行動オントロジは図 3.12，状態オントロジは図 3.13 を用いる．

3.5.1.3　評価項目

本実験の評価項目においても，学習曲線を評価対象とする．また，前節の実験と同様に式 (3.13) の JS，式 (3.14) の RJS，式 (3.15) の DCS，式 (3.16) の RDCS を評価する．

表 3.7 ヘテロジーニアスな MARL を用いた計算機実験における数値結果

Domain	JS	RJS	DCS	RDCS
Source task (Self-transfer)	249.46	0.26	50.14	0.63
Target task (HTL)	4058.09	0.13	2091.87	0.20



(a) Self-transfer　　(b) HTL

図 3.17 ヘテロジーニアスな MARL を用いた計算機実験における学習曲線

3.5.2　実験結果

自己転移の実験結果における，学習曲線を図 3.17(a) に示し，HTL における学習曲線を図 3.17(b) に示す．また，それぞれの JS と RJS，DCS，RDCS を表 3.7 に示す．

本結果の学習曲線は，500 episode 間隔で 10 trial のステップ数平均と標準偏差を用いて描画している．

3.5.2.1　自己転移の実験結果

図 3.17(a) において，明らかな JS と改善された DCS が発現した．JS では 249.46 step であり，RJS は 0.26，すなわち約 74%ものパフォーマンス改善が発現し，多少の曲線の降下（パフォーマンスの改善）が見られるが，その後は横ばいとなり，パフォーマンスを維持している．また，前節の実験と同様に，DCS に関して Self-transfer の学習曲線が，Without tansfer の学習曲線より，低いステップ数を獲得している．DCS は 50.14 step で

あり，RDCS は 0.63 である．したがって，約 37%のパフォーマンスが改善した状態で，学習曲線が収束していると言える．

3.5.2.2　HTL における実験結果

HTL の実験結果においても，図 3.17(b) から，明らかな JS と DCS が発現した．JS では約 87%のパフォーマンス改善が発現し，その後 With transfer の学習曲線は横這となり，パフォーマンスを維持している．DCS では，これまでの実験同様に Without-transfer の学習曲線より高いパフォーマンスを示している．RDCS では約 80%のパフォーマンス改善している．

これらの結果から，ヘテロジーニアスな MARL にける計算機実験において，HTL は従来の転移学習と同様の効果が得られると同時に，ヘテロジーニアスなエージェント間における転移が可能であることが確認された．

3.5.3　考察

3.5.3.1　自己転移における結果の考察

実験結果より，自己転移であるホモジーニアスなエージェント間における，知識の再利用の効果が発現したといえる．しかし，前節の実験と同様に，DCS は Without-transfer より高いパフォーマンスである．すなわち，自己転移により知識を再利用した Target task のエージェントは，Source task にて獲得した知識のパフォーマンスを獲得している．これは，前節の自己転移の考察と同様に，転移時における ANN による知識の近似が原因であると考えられる．しかし，ANN による知識の関数近似により，パフォーマンスが低下したわけではないため，本論文では詳しい議論は行わない．

これらの結果より，転移するために十分な知識が生成できたことが確認でき，また，前節の実験における自己転移の結果が再現したことも言える．

3.5.3.2　HTL における結果の考察

HTL を用いたヘテロジーニアスな MARL において，With transfer の学習曲線に明らかな JS が発現し，知識の再利用の効果が発現した．本実験のセットアップとして，Target task の 3 台のハンターの内，2 台は自己転移と同様に，同一エージェント間における転移学習である．また，その内 1 台はヘテロジーニアスなエージェント間の転移学習である．

一見，3 台中 2 台が自己転移にて効果が発現している通り，有利な環境であると考えられるが，Target task において，捕獲条件は 3 台のハンターが同時に獲物と隣り合う座標に移動することであり，タスクの難易度が高いと言える．唯一自己転移でない，エージェント 3 は再利用するエージェント 1 の知識を基に行動する．さらに，エージェント 3 とエージェント 1 間では，行動集合 A や環境状態集合 S が異なるこれまでの実験条件では一番ヘテロジーニアスなセットアップである．このようなヘテロジーニアスな環境に置いても，HTL は知識の再利用の効果が発現し，Without-transfer に比べて，With transfer の学習曲線に JS が発現したことは，本研究の目的である，ヘテロジーニアスな MARL 間における転移学習に貢献したと言える．

本実験の Target task は，Source task と比較して捕獲が難しいため，学習曲線の絶対的なステップ数が極端に増加している．これにより，DCS の値が低く，RDCS の値が小さい結果となった．図 3.17(b) における Without-transfer の学習曲線は，パフォーマンスの改善が見られ，学習している傾向が現れている学習曲線であるが，学習パラメータのチューニングに課題が残ると考えられる．この現象は，これまでと同様に ANN が原因であると考えられる．しかし，これまでの考察で述べていることと同様に，With transfer のパフォーマンスへの悪影響は見られないため，議論は行わない．

これらの結果より，ヘテロジーニアスな MARL における HTL を用いた知識の再利用の効果が発現したと言える．本実験の結論として，計算機実験であるという前提が必要となるが，ヘテロジーニアスな MARL における転移学習は，HTL により実現可能であることが示唆された．

3.5.3.3　ANN による転移への影響

本実験の結果より，転移する知識を ANN により関数近似しても Target task のジェージェントに悪影響は及ぼさないことが考えられる．これは，以下の理由により直感的な理解とは異なる．

知識を再利用しないエージェントの学習曲線 (“Without transfer”) における収束値と，知識を再利用したエージェントの学習曲線 (“With transfer”) の収束値は大きく異なる．すなわち，エージェントが環境を一から学習しても，知識を再利用したエージェントのパフォーマンスに到達することが出来ていない．知識の再利用の有無には関係なく，環境を学習するのであれば到達するパフォーマンスは同様であると考えられる．したがって，ANN による関数近似は転移先のエージェントに対して，知識の軽量化や形式化以上の効果を与えていることになる．

ANN による関数近似を行う際，獲得した Look-up table の行動価値を次式により 2 値化している（以降 2 値化知識と呼ぶ）.

$$Q(s,a_i) = \begin{cases} 1 & (Q(s,a_i) = \max_{a \in A} Q(s,a)) \\ 0 & (otherwise) \end{cases} \tag{3.17}$$

これは，環境状態の入力に対し，ANN の出力ノードのいずれかを発火させるためである．Look-up table に記されている行動価値を直接 ANN により近似すると，近似誤差により準最適でない行動の選択確率が変動する可能性があるという考えの下に，2 値化処理を行った．ANN により近似した 2 値化知識を再利用する場合，Target task のエージェントは式 (3.18) により 1 が代入された行動を選択しやすくなる．これにより，Target task のエージェントは準最適な行動の選択確率が上昇し，Without-transfer より良いパフォーマンスが発現したと考えられる．しかし，この現象は階層的転移学習のメカニズムによる効果では無く，知識のデータ量削減を目的に行った関数近似の効果であることに注意が必要である．

この ANN によるフィルタ的な動作は，有益な転移手法の一種と成り得るが，階層的転移学習の効果における評価から逸脱するため，上記以上の解析は行わない．

3.6 ヘテロジーニティによる HTL の効果

これまでの実験では，行動のヘテロジ―ニティや観測可能な環境状態におけるヘテロジーニティに着目して HTL の効果を検証した．本実験では，MARL におけるエージェントの組合せや，転移の難易度が異なる 6 種の条件において HTL の効果を検証する．また，自己転移による知識の生成の確認実験も 4 種行う．これにより，少しずつヘテロジーニティの異なる実験条件において，HTL による転移学習が可能か検証する．

本実験においても，シミュレータ・プラットフォームは付録 A に記すマルチエージェント強化学習シミュレータを用い，知識は ANN による関数近似を行う．

3.6.1 問題設定

本実験においても，前節までの実験と同様に追跡問題を用いて評価する．環境設定やエージェントの設定に関して重複する部分も多いため，既に述べられている設定に関しては，引用する形で説明を行う．

3.6.1.1 環境設定

追跡問題のグリッドワールドのサイズは 7× 7 とする．また，グリッドワールドの周囲四辺を飛び越える移動はできない．エージェントはハンターを 2 台用い，獲物を 1 台用いる．ハンターエージェントや獲物の移動特性，視覚範囲は図 3.11 と同様に，3 種類のハンターと 1 種類の獲物を用いる．これまでと同様に，獲物の捕獲条件は 2 台のハンターが同時に隣り合う座標に移動した時とする．各エージェントの行動順序はハンター 1→ ハンター 2→ 獲物とする．捕獲状態に到達したら，各エージェンとは初期位置にリセットされる．各エージェンとの初期座標は，図 3.9(a) と同様である．

3.6.1.2 エージェントのヘテロジーニティの再定義

各ハンターのヘテロジーニティは，グリッドワールド内の移動方向と 1 回の行動における移動距離，行動の種類の数である．各エージェント間のヘテロジーニティについて二項関係で表現すると，移動方向に関しては $Agent1 \neq Agent2$，$Agent1 \approx Agent3$，$Agent2 \approx Agent3$ の関係にあると考えられる．移動距離に関しては，$Agent1 = Agent2$，$Agent1 \approx Agent3$，$Agent2 \approx Agent3$ であり，行動の種類の数では $Agent1 = Agent2$，$Agent1 \neq Agent3$，$Agent2 \neq Agent3$ の関係にあると考えられる．ここでは，ヘテロジーニティの

表 3.8 各エージェント間におけるヘテロジーニティスコア

Agent pair	Direction	Distance	Number of actions	Total score of heterogeneity
Agent1 & Agent2	3 ($\neq$)	1 ($=$)	1 ($=$)	5
Agent1 & Agent3	2 ($\approx$)	2 ($\approx$)	3 ($\neq$)	7
Agent2 & Agent3	2 ($\approx$)	2 ($\approx$)	3 ($\neq$)	7

度合の直感的理解のために，関係演算子 $=$，$\approx$，$\neq$ にそれぞれ近い順に，$=$ は 1，$\approx$ は 2，$\neq$ は 3 のようにポイントを付加する．この数字が大きいほどヘテロジーニティが高いと判断する．各ハンター間のヘテロジーニティの要素とポイント，それらの合計スコア（以降ヘテロジーニティスコアと呼ぶ）を表 3.8 に記す．これにより，エージェント間でどれだけヘテロジーニアスなのか定量的に表現する．本数値は，あくまでもヘテロジーニティの度合いを表すための指標である．ホモジーニアスなエージェント間のポイントとしては，移動方向，移動距離，行動数が全て 1 であるため，合計スコアは 3 とする．

3.6.1.3　実験条件

本実験では，大きく分けて 2 つの実験を行う（図 3.18）．1 つは，Source task で獲得した知識が，正しく生成されているかを検証するために，Source task と同条件の追跡問題へ知識の転移を行う自己転移実験である．言い換えると，自己転移は生成した知識を自分自身で再利用する実験条件である．もう 1 つは，HTL を用いたヘテロジーニアス環境における転移学習実験を行う．3 種のハンターを用いて，そのうち 2 台を MARS として構成し，表 3.9 に示す計 6 種類の条件により計算機実験を行う．本実験条件で対象としているヘテロジーティは 2 つに分類される．1 つは MARS 内，すなわち協調するエージェント間でヘテロジーニアスな場合である．2 つ目は，知識を転移する Source task と Target task 間の MARS でヘテロジーニアスな場合である．実験条件 A はタスク間でヘテロジーニアスであり，各タスクの MARS 内ではホモジーニアスな環境である．実験条件 B は Source task の MARS はホモジーニアスであるが，タスク間や Target task の MARS はヘテロジーニアスである．Target task の Agent 1 は，知識を再利用しながら移動特性の異なるエージェントと協調捕獲行動を行う．実験条件 C は，各タスクの MARS 内でヘテロジーニアスであり，タスク間もヘテロジーニアスな環境である．実験条件の中でも最もヘテロジーニティが高い実験条件である．

表 3.9 において，A-2 など 2 つ目の実験条件は，A-1 の Source task と Target task のエージェント設定を逆にしている．これは，ハンターの組合せによる転移の容易さを検証す

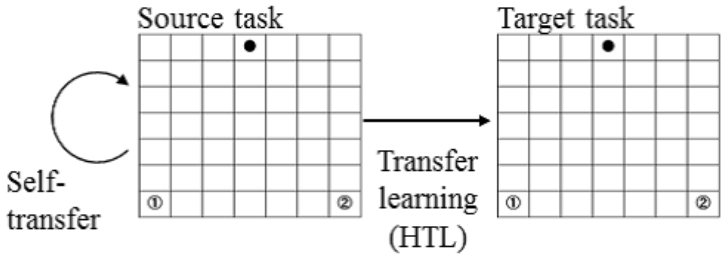


図 3.18 自己転移と HTL を用いた転移学習の実験条件

表 3.9 HTL の実験条件

Class of heterogenety	Experimental condition	Source task Hunter1 Hunter2	Target task Hunter1 Hunter2
Homo-Homo	A-1	Agent 1, Agent 1	Agent 2, Agent 2
	A-2	Agent 2, Agent 2	Agent 1, Agent 1
Homo-Hetero	B-1	Agent 1, Agent 1	Agent 1, Agent 2
	B-2	Agent 1, Agent 2	Agent 1, Agent 1
Hetero-Hetero	C-1	Agent 1, Agent 2	Agent 2, Agent 3
	C-2	Agent 2, Agent 3	Agent 1, Agent 2

るためである．

表 3.8 を基に，各実験条件における Source task と Target task 間のエージェントのヘテロジーニティスコアを，表 3.10 に示す．表 3.10 では，各ハンターで知識を転移する関係にあるエージェント間のヘテロジーニティスコアを記し，さらに合計スコアにより転移の難易度を表している．合計スコアが高ければ，知識の再利用も難しい条件と考えられる．これにより，各実験条件で転移の難しさの関係を直感的に理解可能である．表 3.10 により，各実験条件における転移の難易度を大小関係で表すと次式のように考えられる．

$$\text{Condition C} > \text{Condition A} > \text{Condition B} \tag{3.18}$$

実験条件 A が実験条件 B より難易度が高い理由として，実験条件 B は Source task と Target task でホモジーニアスなエージェントが 1 台存在するからである．

自己転移の実験条件に関しては，表 3.9 における Source task のエージェントの組合せを用いる．すなわち，表 3.11 のように Source task と Target task におけるエージェントの組合せが同じである．これにより，Source task で獲得された知識が正しく作成されているかを確認する．

表 3.10 各実験条件におけるヘテロジーニティスコア

Experimental conditions	Hunter 1	Hunter 2	Difficulty of transfer
Condition A	5	5	10
Condition B	3	5	8
Condition C	5	7	12

表 3.11 自己転移の実験条件

Experimental condition	Source task Hunter1 Hunter2	Target task Hunter1 Hunter2
S-1	Agent 1, Agent 1	Agent 1, Agent 1
S-2	Agent 2, Agent 2	Agent 2, Agent 2
S-2	Agent 1, Agent 2	Agent 1, Agent 2
S-2	Agent 2, Agent 3	Agent 2, Agent 3

いずれの実験条件においても，Souce task において 10000 エピソードの学習を行い，ハンター 1 が獲得した知識は Target task のハンター 1 へ転移される．同様に，ハンター 2 の知識は Target task のハンター 2 へ転移される．Target task においても 10000 エピソードの実験を行う．また，学習には確率的な挙動が含まれるため，各実験条件は 10 回試行し，その平均を結果とする．

知識を転移する際，エージェントが Source task で獲得した知識を直接転移せず，人工ニューラルネットワーク（Artificial neural network : ANN）により関数近似した後に転移する．これは，エージェントの獲得知識の形式は，エージェント内部メカニズムに依存するからである．例えば，学習器や観測可能な状態数，実行できる行動数，行動価値が異なる可能性がある．実世界での利用を考慮すれば，様々なエージェントが，異なる内部メカニズムを持つことは十分に考えられる．したがって本研究では，転移する知識の形式を整えるために本論文における ANN のセットアップとして，入力ノード数=7，中間ノード数=14，出力ノード数=5 に設定し，誤差逆伝搬法を用いて知識の近似を行った．

学習パラメータも含め，本実験で用いるパラメータを表 3.12 に示す．

3.6.1.4　評価項目

本実験の評価項目においても，学習曲線を評価対象とする．また，前節の実験と同様に式 (3.13) の JS，式 (3.14) の RJS，式 (3.15) の DCS，式 (3.16) の RDCS を評価する．

表 3.12 ヘテロジーニティによる HTL の効果実験におけるパラメータ設定

Parameter	Value
Learning rate	0.7
Discount rate	0.9
Reward	1
Boltzmann parameter	0.01
Default Q-value	0
Transfer rate (Self-transfer)	1.0
Transfer rate (HTL)	0.01, 0.1, 1.0
Number of episode of source task	10000
Number of episode of target task	10000
Number of trial	10
Input node of ANN	7
Hidden node of ANN	14
Output node of ANN	5

3.6.2　実験結果

自己転移の実験条件における，S-1 から S-4 の学習曲線を図 3.19 に示し，それぞれの JS と RJS，DCS，RDCS を表 3.13 に示す．HTL を用いた転移学習における実験結果を図 3.20 に示し，JS と RJS，DCS，RDCS を表 3.14 に示す．

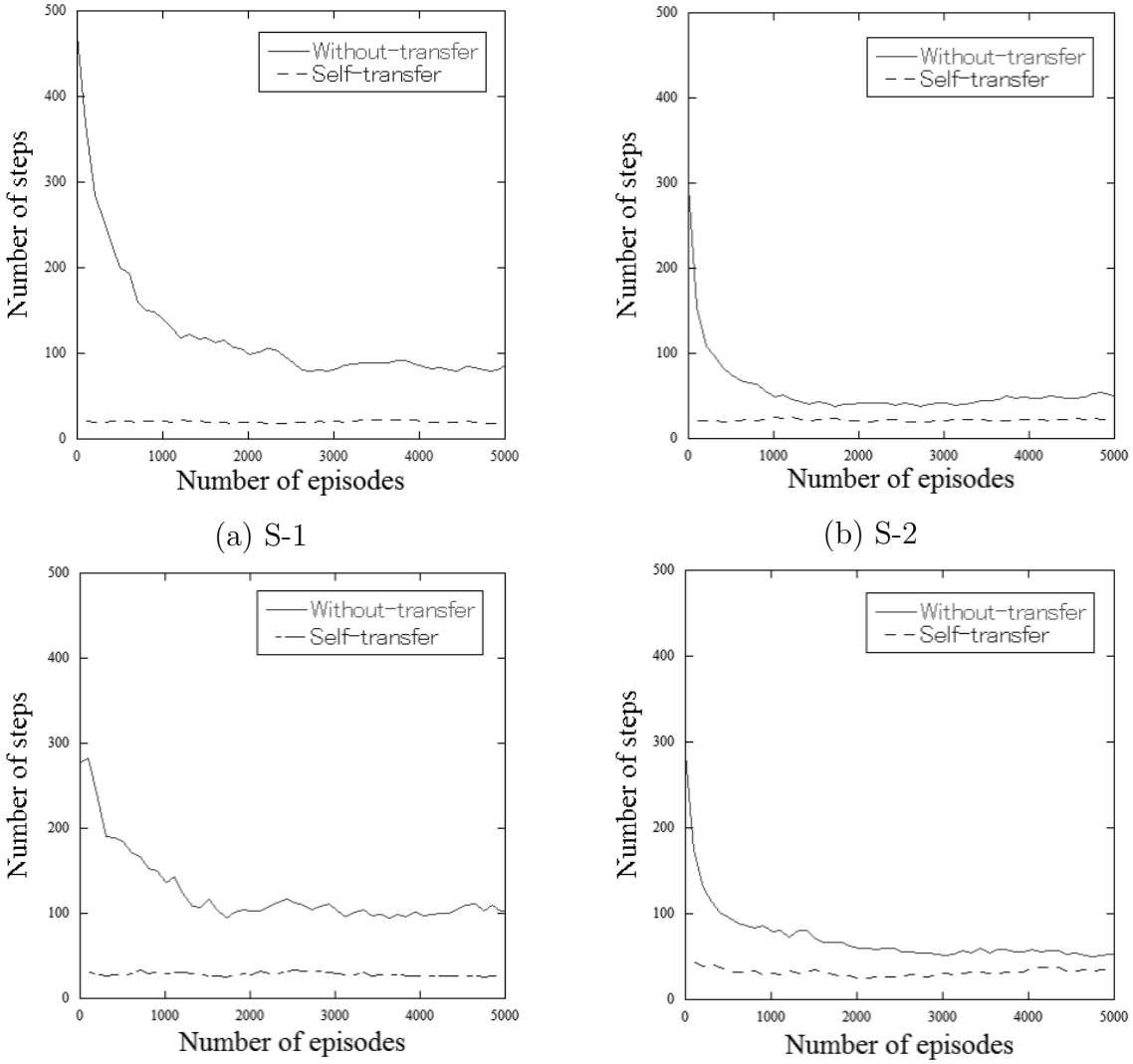
本結果の学習曲線は，膨大なデータの傾向を表すために，付録 B に示すスムージング処理を行いグラフの描画を行っている．

3.6.2.1　自己転移の実験結果

前節までの実験と同様に，各実験条件において JS が発現した．図 3.19（a）～（d）から見てもわかるとおり学習曲線は初期エピソードから低いステップ数となり，そのパフォーマンスを維持したまま横這いの波形となっている．全ての実験条件において，表 3.13 の RJS から，self-transfer が without-transfer より約 80%のパフォーマンスが向上した状態で学習がスタートしていることがわかる．

3.6.2.2　HTL の実験結果

全ての実験条件において，転移率を 0.01 から上昇させると，JS が大きくなることがみてとれる．特に，転移率 1.0 においては，どの実験条件においても，明らかな JS が発現



(a) S-1　　(b) S-2

(c) S-3　　(d) S-4

図 3.19 自己転移における各実験条件の学習曲線

表 3.13 自己転移における JS と DCS

Experimental condition	JS	RJS	DCS	RDCS
S-1	389.11	0.05	55.91	0.27
S-2	139.90	0.15	33.61	0.44
S-3	315.55	0.09	78.93	0.33
S-4	151.76	0.21	23.71	0.60

表 3.14 HTL における JS と DCS ($\tau = 1.0$)

Experimental condition	JS	RJS	DCS	RDCS
A-1	136.17	0.18	31.96	0.47
A-2	386.31	0.13	4.36	0.94
B-1	319.03	0.09	83.11	0.26
B-2	365.38	0.11	41.83	0.46
C-2	143.26	0.25	20.18	0.66
C-2	252.05	0.28	31.61	0.72

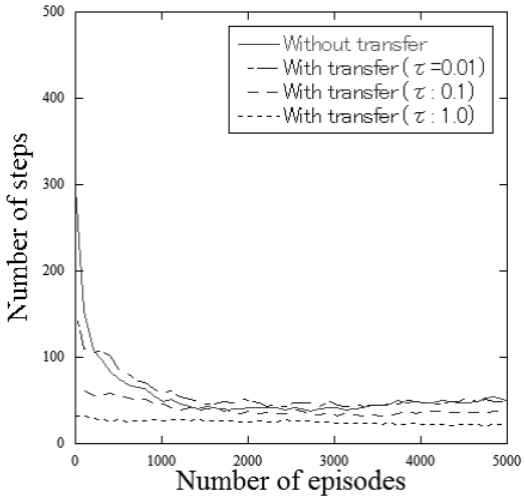
後に自己転移の実験結果同様，横這いの学習曲線となっている．

DCS や RDCS においても，全ての学習曲線において改善されていることがわかる．しかし，自己転移の結果と異なり，RDCS の値は約 0.5〜1.0 に納まっている．

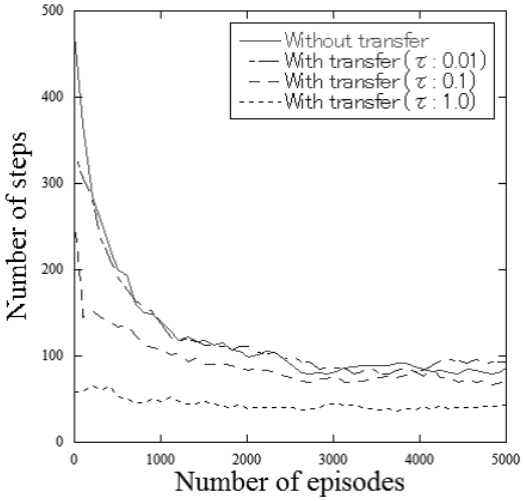
3.6.3 考察

3.6.3.1 自己転移における効果

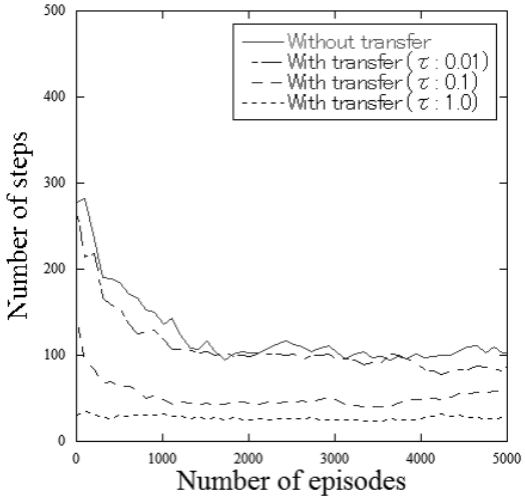
すべての実験条件の DCS において，転移無し（Without-transfer）の学習曲線より低いステップ数が発現している．表 3.13 の RDCS から，自己転移後の学習曲線は，10000 エピソードの学習を繰り返しても，Without-transfer の学習曲線と比較して，40%以上のパフォーマンス改善が現れている．すなわち，自己転移されたエージェントは，Source task のエージェントと比較して，良いパフォーマンスを獲得したこととなる．言い換えると，この結果は強化学習では到達できないパフォーマンスが発現していることを意味する．これは，知識の転移時における ANN による関数近似が原因であると考えられる．一般的に，エージェントが獲得するパフォーマンスは，強化学習器のパラメータチュー



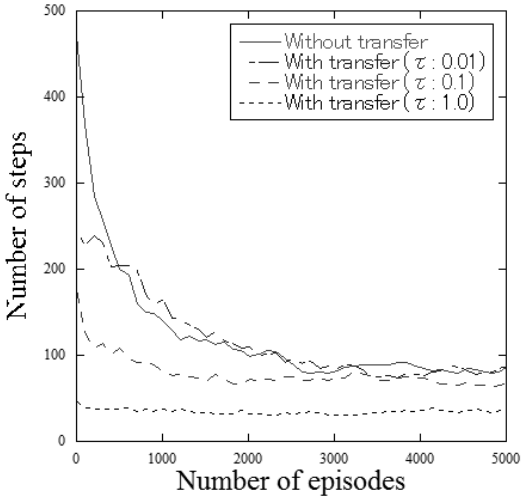
(a) A-1



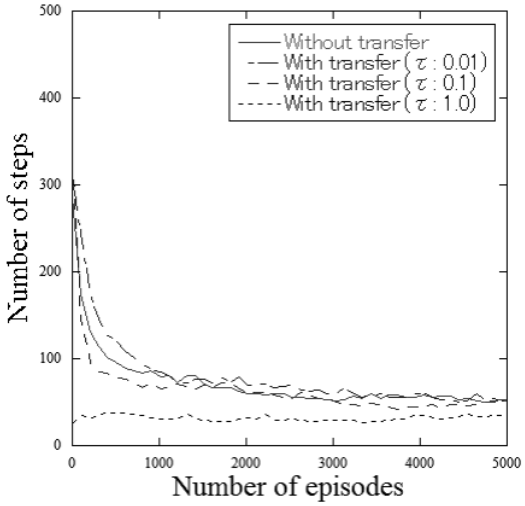
(b) A-2



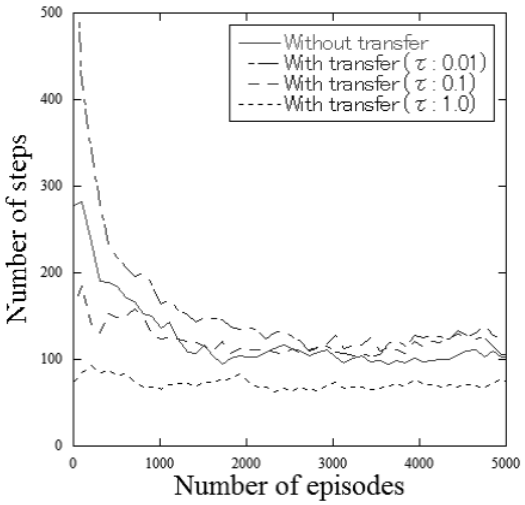
(c) B-1



(d) B-2



(e) C-1



(f) C-2

図 3.20 HTL における各実験条件の学習曲線

ニングに依存する．図 3.19(a)〜(d) の Without-transfer における学習曲線では，学習によるパフォーマンの向上が見られているが，曲線が到達するステップ数は最適でない可能性がある．それらの知識を，ANN により関数近似することで，問題解決に有効な行動が優位に選択されたと考えられる．この効果は前項の考察と同様に，ANN による知識の近似が原因であると考えられる．転移する Source task の知識は，ANN の近似前に式 (3.18) を用いて 2 値化処理されている．ANN による関数近似後は，準最適な行動が選択されやすくなっている．

これらの自己転移の結果から，各実験条件において，HTL による知識の転移をするために十分な知識が獲得できたと言える．

3.6.3.2　HTL における効果

HTL を用いた実験における結果では，自己転移の結果同様に，$\tau = 1.0$ の時全ての学習曲線で，明らかな JS が発現し，OITM を用いた転移学習の効果が現れていると考えられる．また，転移率を高めると，JS が大きくなる傾向も表れた．

HTL の JS と RJS において，実験条件 C の RJS が他の実験条件と比較して，転移の効果が得られていない．これは，3 種の実験条件の中でも，最もヘテロジーニアスな条件であり，Source task にて獲得して協調的な捕獲行動が Target task で再利用しにくい条件であると考えられる．しかし，波形としては，他の実験結果と同様に横這いであり，JS としての効果も十分表れていると言える．

HTL の RDCS では，自己転移の RDCS と比較して，高い値となっている．すなわち，Without-transfer のパフォーマンスより良い知識を獲得しているが，自己転移の実験条件ほどの効果は得られないと考えられる．これは，知識を再利用するエージェントがヘテロジーニアスなためであると考えられる．しかし，A-1 から C-2 の全実験条件において，Without-transfer と比較して With-transfer の最終的なパフォーマンスは高い．この効果の理由も，自己転移の結果と同様に ANN による知識の近似が考えられる．転移する Source task の知識は，ANN の近似前に式 (3.18) を用いて 2 値化処理されている．ANN による関数近似後は，準最適な行動が選択されやすくなっている．これにより，あらかじめ知識を関数近似して転移を行うことで，Source task で獲得した準最適な知識がブラッシュアップされたと考えられる．

これらの結果から，ヘテロジーニアスな MARL 間の転移学習において，OITM を用いた転移学習である HTL により，これまでの転移学習と同様の効果が確認できたと言える．

3.6.3.3　転移の難易度における HTL の効果

ここでは，表 3.10 を用いて表した転移の難易度により実験結果を考察する．各実験条件における，転移の難易度の関係は C>A>B とした．表 3.14 の RJS をみると，上記難易度の関係に応じてジャンプスタートが得られやすくなっていることが見て取れる．結果として全ての実験条件で JS や DCS が得られているが，転移が難しい条件，すなわち知識を転移するエージェント間のヘテロジーニティが高い場合，JS や DCS が発現しにくい傾向が見られた．しかし，JS の発現の割合は転移の難易度だけではなく，タスクの難易度による初期パフォーマンスと収束値の差にも依存すると考えられる．今後も他タスクを用いた検証等が必要である．

3.7 結言

本章では，ヘテロジーニアスなマルチエージェント環境における転移学習と，新たなエージェントがシステムに参加する場合を想定し，他の全エージェントとの Inter-task mapping を実現する手法について議論した．

ヘテロジーニアスなマルチエージェント間の Inter-task mapping 作業を支援する，オントロジを用いた Inter-task mapping である OITM を提案した．また，OITM を用いた転移学習手法として階層的転移学習を提案し，追跡問題を用いた計算機実験により，以下の成果が得られた．

(1) 行動集合 A のみが異なるヘテロジーニアスな MARL において，OITM を用いた HTL により，従来の転移学習の効果である JS の発現を計算機実験により確認した．

(2) 環境状態集合 S と行動集合 A，タスクの難易度が異なるヘテロジーニアスな条件において，OITM を用いた HTL により，従来の転移学習の効果である JS の発現を計算機実験により確認した．

(3) 知識の関数近似を用いた階層的転移学習において，負の転移は発生せず従来手法通り転移学習の優位性（JS など）が得られることを確認した．

上記の成果より，HTL を用いことで，従来議論が深く行われていないヘテロジーニアスな MARL における転移学習が可能であることを示した．また，ヘテロジーニアスなエージェントが新たにシステムに参加しても，OITM によるオントロジとの Inter-task mapping を定義するこで，全てのエージェントの知識の再利用が可能であることが示唆された．

第4章　知識の再利用時における転移率の検討

4.1　緒言

本章では，ヘテロジーニアスなマルチエージェントの転移学習において，転移する度合いを調整するための転移率というパラメータに関する議論を行う．表 1.1 に示した通り，本章は 1.4 節に提示した以下の 議論点に対してアプローチを行う議論の 1 つである．

- ヘテロジーニティに関する議論

これまでの転移学習では，式 (2.7) のように Source task の知識と Target task の知識を結合していた [40]．近年のアプローチとしては，高野は Source task の知識と Target task の知識を加重平均により結合する手法を提案している [55]．また本論文では，前節にて Source task の転移度合いを調節するパラメータ τ を用いた．しかし，これら “転移する度合いを調整するパラメータ” が，転移学習に対してどのような効果をもたらすのか議論が不足している．

ヘテロジーニアスなマルチエージェントにおける転移学習を考慮すると，単純に知識を転移することが有用とは限らないと考えられる．エージェントのヘテロジーニティによっては，本論文で転移率と呼ぶパラメータを調整し，転移度合いを調整して知識の再利用を行うことが必要であると考える．したがって，以下の課題を設定する．

課題：「転移学習における転移率の効果に関する検討が必要」

本章では “転移する度合いを調整するパラメータ” を転移率と統一して呼ぶこととし，静的環境や動的環境の転移学習において，転移率が転移学習に対してどのような効果をもたらすのか議論する．また，ヘテロジーニアスなエージェントにおける転移学習において，どれくらいの値を転移率として用いればよいか，計算機実験により明らかにする．

4.2　記号の準備

これまでの転移学習に関する研究において，再利用知識や転移率などの記号は各自定義し，述べられてきた．本節では，使用する記号による混乱を避けるため，転移学習における転移率の検討を行う前に，本章で用いる記号を定義しておく．

本章では主に，Source task から転移される知識 $Q^s(s,a)$ と，Target task で新たに学習する知識 $Q^t(s,a)$，それらが演算子により結合された知識 $Q^c(s,a)$ を用いる．それぞれの知識は，添え字の s,t,c により区別する．また，s や a はそれぞれのタスクにて観測された環境状態と，それに対応する行動を意味する．記述の単純化のため，転移される知識 $Q^s(s,a)$ の記述には，ITM の関数 $\chi_S(\cdot)$ や $\chi_A(\cdot)$，OITM の $\chi_S^O(\cdot)$，$\chi_A^O(\cdot)$ は用いない．$Q^s(s,a)$ は ITM 等の処理が含まれているものとする．これらの標記を表 4.1 に示す．

表 4.1 転移率の議論における記号

Symbol	Descreption
$Q^s(s,a)$	: Transferred knowledge (Knowledge of source task)
$Q^t(s,a)$	: Knowledge of target task
$Q^c(s,a)$	: Combined knowledge

4.3　現在の知識と再利用知識の結合

エージェントが Target task にて知識を再利用するために，$Q^t(s,a)$ と $Q^s(s,a)$ を結合する手法がいくつか提案されている．本論文では，知識を結合する式を知識結合法と呼ぶ．次項から知識結合法と，それらの問題点を述べる．

4.3.1　既存の知識結合法

Tyalor の転移学習では，Source task の知識を再利用する知識結合法として，次式を用いている．

$$Q^c(s,a) = Q^t(s,a) + Q^s(s,a) \tag{4.1}$$

式 (4.1) は，転移率を用いない最も単純な形の知識結合法である．これに対し，Takano らは転移度合いを調整するパラメータ ζ を導入し，さらに，知識同士の結合に加重や平

均を用いた手法も提案している [55, 75–77]．それらの知識結合法を次式から示す．

$$Q^c(s,a) = Q^t(s,a) + \zeta Q^s(s,a) \tag{4.2}$$
$$Q^c(s,a) = \frac{1}{2}\{(1-\zeta)Q^t(s,a) + \zeta Q^s(s,a)\} \tag{4.3}$$
$$Q^c(s,a) = (1-\zeta)Q^t(s,a) + \zeta Q^s(s,a) \tag{4.4}$$

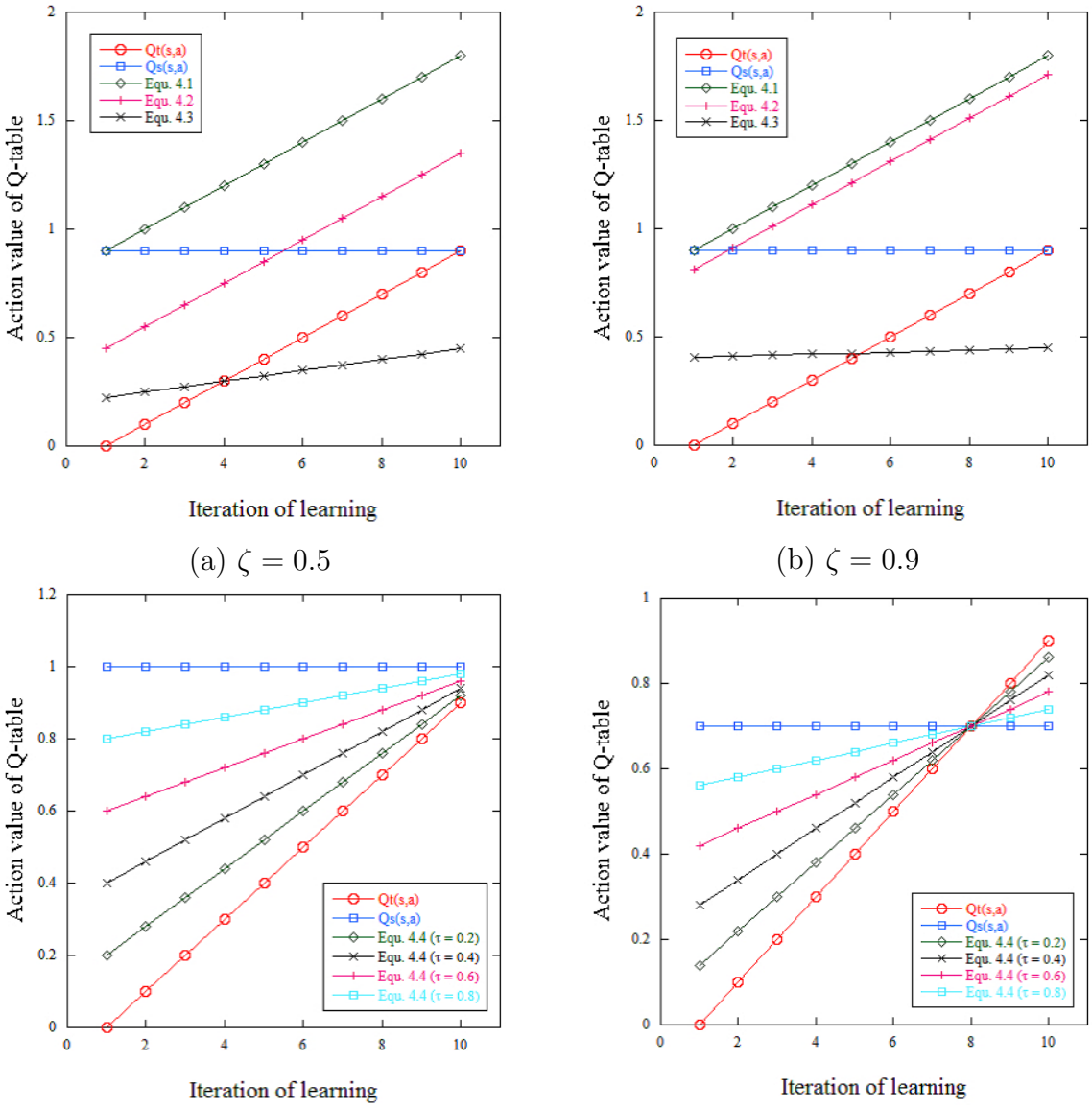
これらの式は，荷重となる転移率 ζ を $Q^s(s,a)$ にのみ与えている．エージェントは，Target task にて常に $Q^s(s,a)$ を活用する．Takano らの論文では，それぞれの手法を評価し有用性を示している．また，式 (4.2) の有用性は，前章に示した実験結果の通りである．

4.3.2　知識再利用法の問題点

式 (4.1) は最も単純な形式であり，Boutsioukis らの研究などでも採用され, 有用性が示されていると言える [41]．しかし，式 (4.1) や式 (4.2) は，過去の知識と現在の知識の単純和であるため，例えば $Q^s(s,a)$ が非常に高い行動価値を持っていたとすると，Target task においてエージェントが Source task の知識を基に行動するため，新たな環境に適応できず過学習に陥る可能性が考えられる．したがって，式 (4.1) や式式 (4.2) は，結合した知識の行動価値が高くなる．ここで，$Q^s(s,a)$ は十分に学習した行動価値として 0.9 とし，$Q^t(s,a)$ が $0.1 \sim 0.9$ と変化させた時における式 (4.1) や式式 (4.2)，式 (4.3) に示した各知識結合法により計算される行動価値 $Q^c(s,a)$ の変化を図 4.1(a)，(b) に示す．

図 4.1(a)，(b) において，式 (4.1) や式 (4.2) は $Q^t(s,a)$ の行動価値が $Q^s(s,a)$ に近づくにつれて，$Q^c(s,a)$ も高くなるが $Q^t(s,a)$ や $Q^s(s,a)$ のそれぞれの行動価値をはるかに上回る値を取りうる．さらに，式 (4.3) は，Target task で学習が十分に進み $Q^t(s,a) = Q^s(s,a)$ となっても行動価値は $\frac{1}{2}$ なる．Target task で獲得した有用な $Q^t(s,a)$ が活用できていない．式 (4.4) の転移率に対する変化を図 4.1(c)，(d) に示す．式 (4.4) においては，$Q^c(s,a)$ が $Q^t(s,a)$ と $Q^s(s,a)$ 間の値を取りうる．しかし，式 (4.4) もこれまで述べた知識結合法と同様に，ζ は任意の値の固定値である．すなわち，Target task における学習が進み，十分に $Q^t(s,a)$ の行動価値が高くなっても，常に $Q^s(s,a)$ が参照され続ける．この場合，学習初期の段階においては何かしらの方策に則った行動を起こすことは，ランダムな行動より得策であると考えられる．しかし，新たな環境 (Target task) において，十分に学習した場合は再利用知識は必要としない．そのため，Target task の学習が十分に進んだ場合は再利用する知識 $Q^s(s,a)$ を使用しない知識結合式が必要であると考える．

そこで次項に，Target task で知識を再利用する際に学習進度に応じて再利用知識の度



(a) $\zeta = 0.5$　　(b) $\zeta = 0.9$

(c) $Q^t(s,a)$ が $Q^s(s,a)$ より低い場合　　(d) $Q^t(s,a)$ が $Q^s(s,a)$ を上回る場合

図 4.1 行動価値の変化に対する結合知識の行動価値

合いを調整する知識結合法を提案する．

4.3.3　学習進度に応じた知識の再利用度合調整

本研究では，Target task の学習進度に応じて，知識 $Q^s(s,a)$ の再利用度合を調整する知識結合法を提案する．本研究で提案する知識結合法は，以下の 2 つの機能を実現する．

- Target task において $Q^s(s,a)$ に任意の τ を付加し，行動価値を割り引く．
- Target task で十分に学習した場合，$Q^s(s,a)$ を利用しない．

これらの機能を実現する知識結合法を次式に示す．

$$\begin{aligned} Q^c(s,a) &= Q^t(s,a) + \tau\delta Q^s(s,a) \\ \delta &= R_f - Q^t(s,a) \end{aligned} \tag{4.5}$$

ここで，$\tau(0 < \tau < 1)$ は任意の定数であり，R_f はタスク達成時にエージェントが獲得できる報酬である．一般的に強化学習で記される r と同義である．δ は行動価値が収束する報酬 R_f の値と $Q^t(s,a)$ の差である．すなわち，δ は報酬までの残差であり，学習進度を表す．

式 (4.5) を用いる場合，ある時間 t における結合知識を $Q^c_t(s,a)$ とし，知識の時系列における遷移を考える．この時，学習開始時間 $(t \to 0)$ と十分に学習した時 $(t \to \infty)$ は次式となる．

$$\begin{aligned} \lim_{t\to 0} Q^c_t(s,a) &= 0 + \tau \cdot (1-0) \cdot Q^s(s,a) \\ &= \tau Q^s(s,a) \end{aligned} \tag{4.6}$$

$$\begin{aligned} \lim_{t\to \infty} Q^c_t(s,a) &= Q^t_\infty(s,a) + \tau \cdot 0 \cdot Q^s(s,a) \\ &= Q^t_\infty(s,a) \end{aligned} \tag{4.7}$$

ここで，$Q^t_0(s,a)$ すなわち初期値は 0，報酬 $r = 1$ と仮定している．また，十分に学習した知識の行動価値は報酬 R_f に収束することを前提とする [26]．再利用する知識 $Q^s(s,a)$ は，Target task において改善は行われないこととする．すなわち時間 t に依存しない．

学習が進めば δ は小さくなり $Q^s(s,a)$ の値を低くする．これにより，学習初期においては $Q^s(s,a)$ を参照してエージェントは行動選択を行い，十分に学習した状態においては，新たに獲得した $Q^t(s,a)$ を基に行動を選択する．本研究では，$\tau\delta$ も転移率と呼ぶこととする．

注意が必要なのは，本来強化学習において報酬 r は環境から与えられるものであり，エージェントは r の値をあらかじめ知ることはできない．しかし，実際に強化学習を利用する場面においては，r はエージェントや環境の設計者により設定され，既知であると言える．本手法においても報酬の値は既知であることを前提とする．

4.4　静的環境における転移率の効果

本実験では，式 (4.5) の効果を検証するための基礎的実験として，静的環境における計算機実験を行う．また，本実験の静的環境は Source task と Target task で環境が異なる．すなわち，タスクの目的は同様であるが解法が異なる環境設定である．本実験の目的は，転移率の調整に対する転移学習の効果を検証することである．前節の提案手法の有用性を明らかにするために，Takano らが提案した式 (4.4) も用いて性能の対比を行う．以降，Takano らの手法は既存手法と呼ぶ．

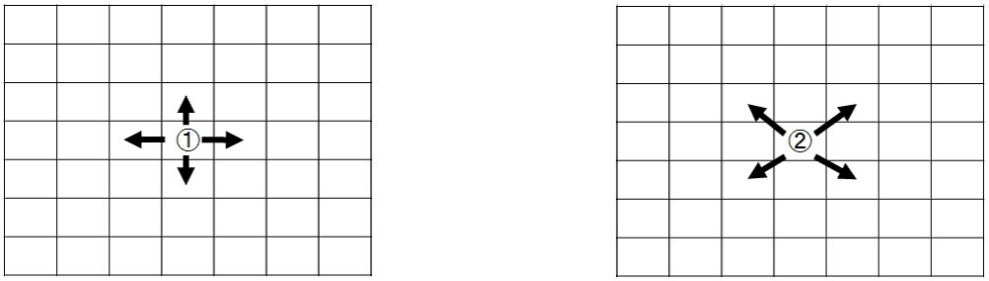
計算機実験を行うシミュレータ・プラットフォームとしては，付録 A のマルチエージェント強化学習シミュレータの設定を変更し，最短経路問題により評価を行う．前章の実験同様，数値シミュレーションを行う．

4.4.1　問題設定

最短経路問題は，グリッドワールド内を強化学習エージェントが，ゴールまでの最短経路を学習するゲームである．また，強化学習エージェントは自己位置を認識可能で，環境の次状態への遷移確率が，強化学習エージェントのみに依存する．したがって，本実験の環境は MDP となる．本節の実験は，第 3 章の追跡問題をベースに問題設定を行う．以下から，問題設定の詳細を述べる．

4.4.1.1　環境設定

本実験では，これまでの実験と同様に 7× 7 のグリッドワールドを用いる．配置する強化学習エージェントは 1 台である．エージェントの初期座標は，ゴール地点から最も遠い場所とする．エージェントがゴールに到達した時，報酬を獲得できる．学習結果として，エージェントは初期座標からゴールまでの最短経路を学習する．ゴールに到着したエージェントは，エピソードが終了し初期座標へ戻る．



(a) エージェント 1　　(b) エージェント 2

図 4.2 最短経路問題で使用する強化学習エージェント

4.4.1.2　エージェント設定

本実験では，図 4.2 に示す 2 種類のエージェントを用いる．それぞれの行動集合は，式 (3.7) の A_{agent1} と A_{agent2} と同様である．したがって，エージェントのヘテロジーニティは行動集合 A の違いである．各エージェントの行動の違いを次式となる．

$$A_{agent1} = \{forward,\ backward,\ right,\ left,\ stop\} \tag{4.8}$$

$$A_{agent2} = \{forward\,right,\ backward\,right,\ forward\,left,\ backward\,left,\ stop\} \tag{4.9}$$

観測可能な環境状態集合は，エージェント 1 とエージェント 2 は共通であり，次式で定義する．

$$S = \{\text{self-location}\} \tag{4.10}$$

式 (4.10) は，自己位置のみ認識可能であることを意味する．本実験の最短経路問題は，他のエージェントが存在せず，ゴールも移動しない．観測すべきものが自己位置だけであるため，各エージェントは視界を持たない設定とした．

ここで，行動集合 A_{agent1} と A_{agent2} は図 4.2(a), (b) の示す矢印方向と対応し，上矢印を $forward$ とする．

4.4.1.3　実験条件

本実験では，図 4.3 に示し，ヘテロジーニアスなエージェント間における転移学習の実験を行う．Source task ではエージェント 1 を用い，Target task ではエージェント 2 を用いる．この実験条件に対して，転移率 τ を 0，0.01，0.1〜1.0（0.1 間隔）で変更し，転移学習を行う．

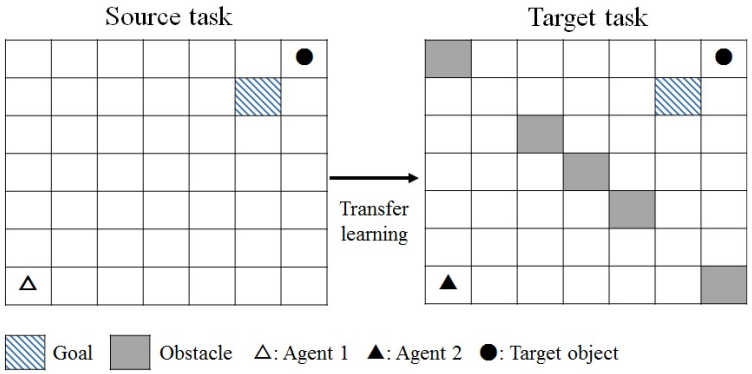


図 4.3 最短経路問題における実験条件

表 4.2 静的環境における転移率の効果のパラメータ設定

Parameter	Value
Learning rate α	0.1
Discount rate γ	0.9
Reward r	1
Boltzmann parameter T	0.01
Default Q-value	0
Transfer rate τ and ζ	0.1, 0.2, $\cdots$, 0.8, 0.9
Number of episode (Source task and target task)	10000
Number of trial	10

本実験では ANN による知識の近似を行わない．したがって，本実験における自己転移は，転移学習が最も効果的に機能する実験条件であり，その実験条件において転移率の変更による影響を調べることで，ヘテロジーニアスな転移学習における転移率の影響を考察することが可能である．

ここで，学習パラメータや実験エピソード数などの値を表 4.2 にまとめる．また，本実験で行う転移学習は，HTL を用いず Taylor らの転移学習を行う．エージェント 1 とエージェント 2 の ITM は図 4.4 のように定義した．

4.4.1.4　評価項目

本実験において，学習の結果として得られる学習曲線を基に転移曲線を評価する．学習曲線は，縦軸にエージェントが目的を達成するまでに要した行動回数，横軸に学習の

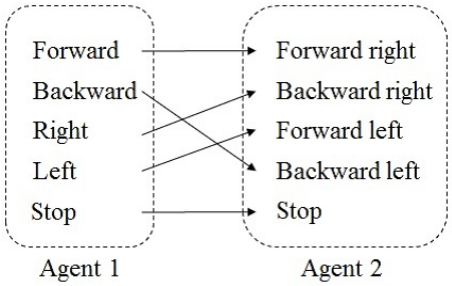


図 4.4 エージェント 1 とエージェント 2 の ITM

繰り返し回数を示したグラフであり，方策の獲得や学習進度を確認することが出来る曲線である．転移曲線は，縦軸に学習初期のステップ数，横軸に転移率をとるグラフの曲線である．転移率の変化に対する，エージェントの初期ステップ数（パフォーマンス）を確認することが出来る．

評価の指標として，各実験で獲得できる学習曲線の初期ステップ数（10 エピソード平均）と収束値（10 エピソード平均）の 2 つを用いる．それらの値を転移率に対してプロットすることで転移曲線と収束値曲線を構成する．初期ステップ数は，知識の再利用を行うことで “知識を再利用せず一から学習” するよりも学習の初期から高いパフォーマンスを発揮しているかどうかの指標である．収束値は，転移学習において初期ステップ数の低下が現れていても，本来収束するはずのステップ数に収束するか評価する指標である．本実験の Target task では，10000 エピソードの実験を行うため最終 10 エピソードの平均値を収束値とし，既存手法と提案手法で比較する．

本実験は MDP であるため，エージェントが獲得すべき最適解が予め計算可能である．最適解は最短経路，すなわり最短ステップ数によるゴールの到着である．図 4.2(a) のエージェント 1 の移動特性から，最短ステップ数はマンハッタン距離により計算できる．n 次元空間における，ある座標 $p = (p_1, p_2, ..., p_n)$ からある座標 $q(q_1, q_2, ..., q_n)$ までのマンハッタン距離 $dm(p, q)$ は次式で定義される．

$$dm(p, q) = \sum_{k=1}^{n} |p_k - q_k| \tag{4.11}$$

さらに，本実験の最短経路問題は 2 次元平面であるため，スタート地点の座標 $p(x_p, y_p)$ からゴール地点の座標 $q(x_q, y_q)$ のマンハッタン距離は次式となる．

$$dm(p, q) = |x_p - x_q| + |y_p - y_q| \tag{4.12}$$

したがって，10 ステップが最短ステップ数となる．

図 4.2(b) のエージェント 2 の移動特性では，斜め移動が可能なため，最短経路はマンハッタン距離ではない．斜め移動における最短経路はチェビシェフ距離により測定可能であるが，本実験条件では障害物が存在しているため，最短経路がチェビシェフ距離とならない．障害物をよけながら移動する最短経路をカウントすると，9 ステップとなる．

4.4.2　実験結果

本実験の結果である転移曲線や収束値曲線は，10 trial の平均値と標準偏差のエラーバーを用いて描画する．

4.4.2.1　初期ステップ数における効果

転移率の変更に対する，学習曲線の初期ステップ数の変化を図 4.5 に示す．初期ステップ数は，実験の 10 回試行における平均とそれぞれの標準偏差を示している．図 4.5 中において，転移率=0 は知識の転移がない学習曲線における初期ステップ数である．したがって，転移率=0 のステップ数からどれだけステップ数が低下するかで知識の再利用における効果を見ることが出来る．これをジャンプスタートと呼ぶ．

図 4.5 から，既存手法と本論文における提案手法は転移率を上昇させるほど，ジャンプスタートが得られることが見て取れる．初期ステップ数という指標において，Takano らの既存手法と比較して提案手法は転移率の効果が損なわれず同等の性能を有していると考えられる．すなわち，知識の再利用によるジャンプスタートを獲得したい場合は，転移率を高めに設定すれば良いことが見て取れる．

4.4.2.2　収束値における効果

転移率の変更に対する，学習曲線の収束値の変化を図 4.6 に示す．収束値は，転移曲線と同様に 10 回の試行における平均とそれぞれの標準偏差を示している．転移率 $= 0$ において収束値は 9 step に収束している．これは，一から環境を学習し収束値に到達するまで episode 数が必要であるが，最適解を獲得していると言える．これは最短経路（最短ステップ数）であり，他の結果の基準となる．

既存手法は転移率 $= 0.1$ から 0.5 までは知識を再利用しジャンプスタートが発現すると同時に，最適解への収束が見て取れる．しかし，既存手法において転移率 $= 0.6$ 以上は収束値が高くなり，また標準偏差も広がっていることが図 4.6 から見て取れる．

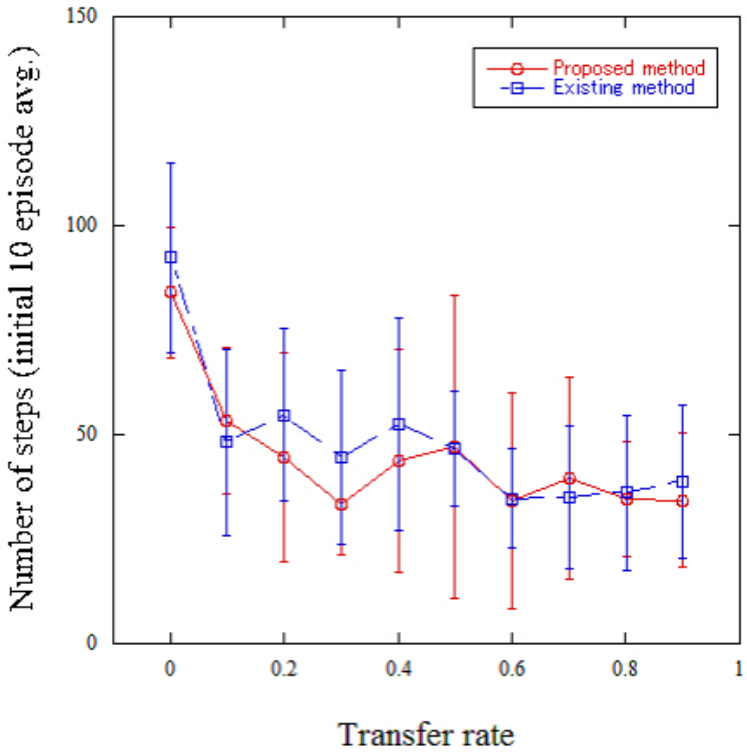


図 4.5 最短経路問題における転移曲線

これに対し，提案手法は転移率を変化させても最適解への収束が発現している．既存手法は設定した転移率に対して，再利用する知識がその分バイアスとして新たに学習する知識に加算される．提案手法は，学習が進み Target task で学習し，十分な行動価値が知識に反映されれば再利用知識を参照しなくなる．これが図 4.6 における，転移率の変化に対する収束値の差が現れた理由だと考えられる．

4.4.3　考察

4.4.3.1　静的環境の初期ステップ数に対する考察

既存手法と提案手法を比較すると，どちらもジャンプスタートが発現することを確認した．これは，どちらの手法も知識の再利用により，学習初期から高いパフォーマンスが発現することを意味する．

図 4.5 を見てもわかるとおり，多少の分散のバラつきはあるが，転移率を高めていくと学習初期のステップ数が低下するため，結論として Target task のエージェントがジャンプスタートを得たい場合は，転移率は高めに設定すると良いことがわかる．

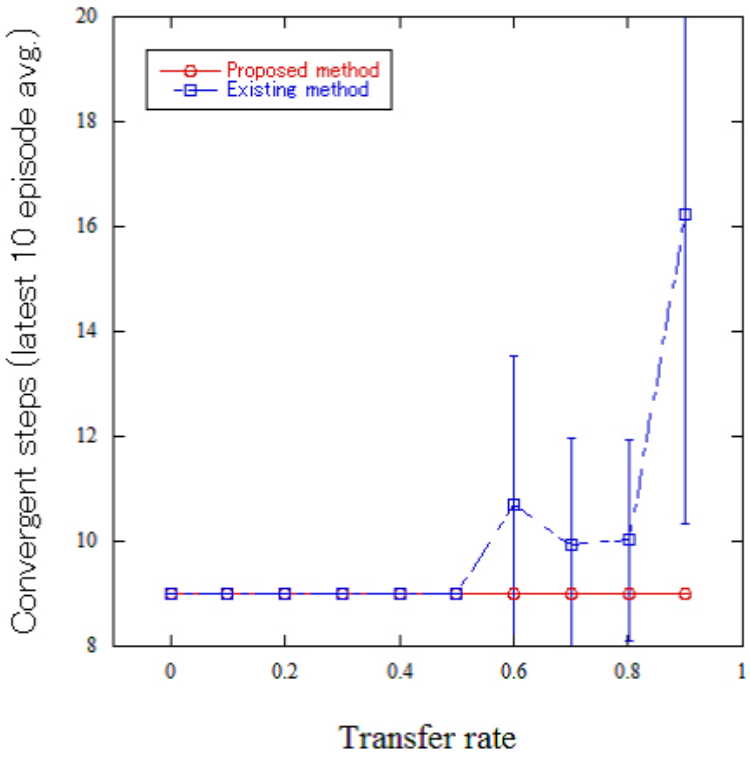


図 4.6 静的環境における収束値曲線

4.4.3.2　静的環境の収束値に対する考察

実験結果から，既存手法は転移率を上昇させると収束値に分散が生じる．転移率が高い場合 ($\tau = \zeta = 0.9$) の学習曲線の比較を図 4.7 に示す．Without transfer は，知識の再利用していないエージェントの学習曲線であり，一から学習した通常の強化学習である．図 4.7 は転移率 $= 0.9$ の場合の学習曲線比較であるが，既存手法 (図 4.7 中の Existing method) は提案手法 (図 4.7 中の Proposed method) と同様のジャンプスタートが発現しているが，静的環境にも関わらず一意の値に収束せず振動が発生している．図 4.7 では 200 episode まで学習曲線を表示しているが，既存手法の振動は増減はあるが 10000 episode まで継続していることを確認している．これは，3.2 節で述べた通り転移率が高いと再利用知識の行動価値が高くなり，学習が進み新たな環境 (Target task) で獲得した知識の利用を妨げているためと考えられる．

これに対し，本論文の提案手法は Target task で学習し，知識の行動価値が高くなると再利用する知識の行動価値を下げるため，転移率を高くしても Target task にて獲得している知識を活用し，環境に対して適応的に行動できると考えられる．さらに，他の学習曲線に比べて早く収束値へ近づいている．

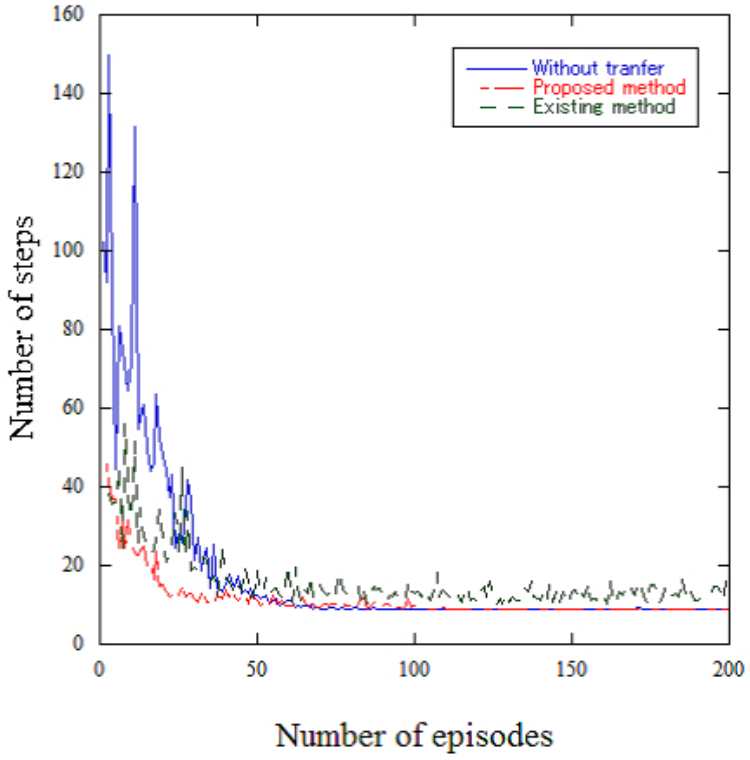


図 4.7 転移率 $\zeta = \tau = 0.9$ の時の学習曲線

これらの結果として，従来手法はジャンプスタートを得るために転移率を高く設定することは，本実験環境において必ずしも効果的でなく調整が必要である．これに対し本提案手法は，転移率によらずジャンプスタートの発現と最適解の獲得に貢献できることが計算機実験により示唆された．

4.5　動的環境における転移率の効果

本実験では，前節で有用性を示した提案手法（式 (4.5)）を動的環境にて検証する．前節の実験では，Source task と Target task でエージェントとタスクがヘテロジーニアスであった．本実験ではさらに，動的環境とエージェント数の変化の設定を追加した追跡問題により検証する．

本実験においても，シミュレータ・プラットフォームとしては付録 A のマルチエージェント強化学習シミュレータを用い，最短経路問題により評価を行う．前節と同様に数値シミュレーションを行う．

4.5.1　問題設定

本実験では，第 3 章で採用してた追跡問題を採用する．動的環境を再現するために，複数台の学習可能なハンターエージェントと，その行動から逃避する獲物エージェントを用いる．

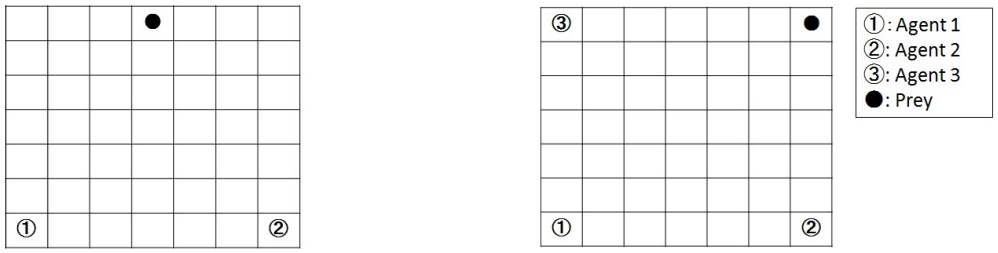
評価として，前節の実験と同様に転移率に対するジャンプスタートの発現と，収束値を評価する．

4.5.1.1　環境設定

本実験においても，環境として 7× 7 のグリッドワールドを用いる．配置する強化学習エージェントであるハンターは 2 台または 3 台とし，獲物を 1 台グリッドワールドに配置する．

本実験でも，第 3 章の実験と同様に Source task と Target task で追跡問題の設定が異なる．Source task ではハンターを 2 台，獲物を 1 台用いる（図 4.8(a)）．Source task は 2 vs. 1 タスクである．Target task においては，ハンターを 3 台，獲物を 1 台用いる（図 4.8(b)）．Target task は 3 vs. 1 タスクである．これにより，Source task と Target task で観測可能な環境状態がヘテロジーニアスとなる．また，後述するが Source task と Target task でエージェントもヘテロジーニアスに設定する．

各エージェントの各タスクにおける初期座標は図 4.8 の通りである．捕獲条件は第 3 章の図 3.10 と同じである．



(a) Source task における初期位置　　(b) Target task における初期位置

図 4.8 追跡問題のエージェント初期位置

表 4.3 動的環境における転移率の効果実験に対するエージェント割当

Task	Srouce task		Target task		
Agent number	Agent 1	Agent 2	Agent 1	Agent 2	Agent 3
Hunters	Hutner 1	Hunter 1	Hunter 1	Hutner 2	Hunter 3

4.5.1.2　エージェント設定

本実験のエージェントは，前節と異なり追跡問題であるため，第 3 章の図 3.11 に示した視界を持つ 3 種類の異なる移動特性を持つエージェントを用いる．

4.5.1.3　実験条件

Source task と Target task で用いるハンターを，表 4.3 のように設定する．各タスクのエージェントにおける知識の転移方向として，Source task の Agent 1 である Hunter 1 の知識は，Target task の Agent 1 である Hunter 1 に転移する．Source task の Agent 2 である Hunter 1 の知識は，Target task の Agent 2 である Hunter 2 と，Agent 3 である Hutner 3 に転移する．

本実験では，上述の通り各タスク間で行動がヘテロジーニアスなエージェントを用い，タスクも異なるため，エージェントの観測可能な環境状態もヘテロジーニアスである．また，前節の実験同様に ANN による知識の近似は行わない．

本実験の強化学習パラメータや実験エピソード数などは，表 4.4 とする．さらに，エージェント 1 とエージェント 2 の ITM も図 4.4 と同様である．本実験で追加されたエージェント 3 においては，エージェント 1 とエージェント 3 の ITM を図 4.9 のように設定した．また，Source task と Target task の環境状態の ITM は図 4.10 のように設定した．図 4.10 において，Target task のハンターは得られる環境上の中から x-coordinate of third hunter

表 4.4 動的環境における転移率の効果のパラメータ設定

Parameter	Value
Learning rate α	0.1
Discount rate γ	0.9
Reward r	1
Boltzmann parameter T	0.01
Default Q-value	0
Transfer rate τ	0.1, 0.2, $\cdots$, 0.9, 1.0
Number of episode (Source task and target task)	10000
Number of trial	10

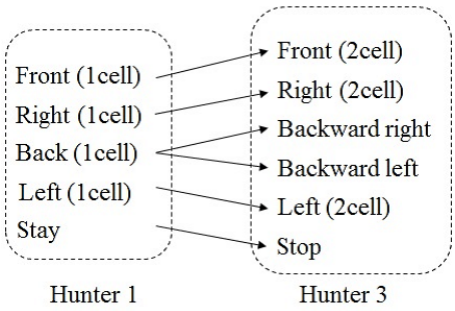


図 4.9 ハンター 1 とハンター 3 の行動の ITM

と y-cordinate of third hunter の情報以外を，Source task のハンターが獲得した知識に入力することを意味する．Srouce task でハンターが獲得した協調捕獲行動を行う知識に対して，このマッピングにより，その知識を再利用する Targe task のハンターがどのハンターと協調捕獲行動を行うかを指定することが出来る．本実験では，ハンター 1 の Second hunter はハンター 3 とし，ハンター 3 の Second hunter はハンター 1 とする．ハンター 2 の Second hunter はハンター 1 とした．

4.5.1.4　評価項目

本実験においても，前実験と同様に学習曲線を基に転移曲線と収束値曲線を評価する．各実験で獲得できる学習曲線の初期ステップ数（10 エピソード平均）と収束値（10 エピソード平均）の 2 つ転移率に対して評価する．

本実験は，マルチエージェントかつ獲物が逃避行動を行うため動的な環境である．し

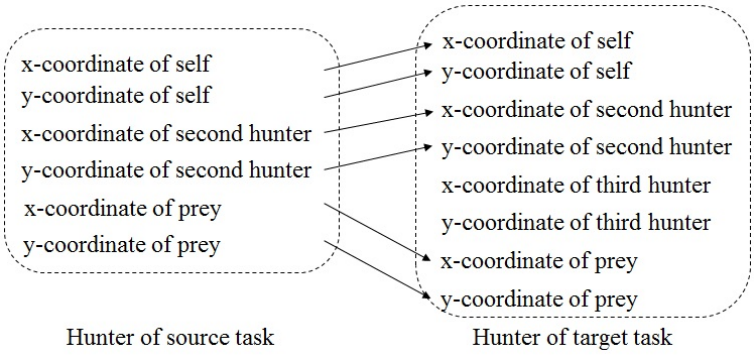


図 4.10 Source task と Target task の環境状態の ITM

たがって，前実験のマンハッタン距離などのように最適解を計算することができない．学習曲線におけるステップ数の減少傾向により，最適解の獲得傾向を判断する．

4.5.2　実験結果

本実験においても前節の実験結果と同様に，学習曲線は 10 trial の平均と標準偏差を用いて描画する．

4.5.2.1　初期ステップ数の結果

転移率の変更に対する各学習曲線の初期ステップ数を図 4.11 に示す．本実験においても，転移率 $= 0$ は転移しない学習曲線における初期ステップ数である．すなわち，このステップ数を基準にして転移率を上昇させるとどのような効果が発現するか確認する．

図 4.11 から，転移率を上昇させると初期ステップ数は低下し，高いジャンプスタートが発現することが見て取れる．また，転移率がおおよそ 0.4 からジャンプスタートが一定となることが見て取れる．

4.5.2.2　収束値の結果

転移率の変更に対する，各学習曲線の収束値の変化を図 4.12 に示す．転移曲線と同様に，転移率 $= 0$ の時のステップ数を基準にして各転移率における収束値を評価する．

本実験の収束値曲線では，転移曲線と同様に転移率の上昇に伴いステップ数の減少傾向が発現した．これは，知識の転移無しで一から学習し，獲得した知識よりパフォーマ

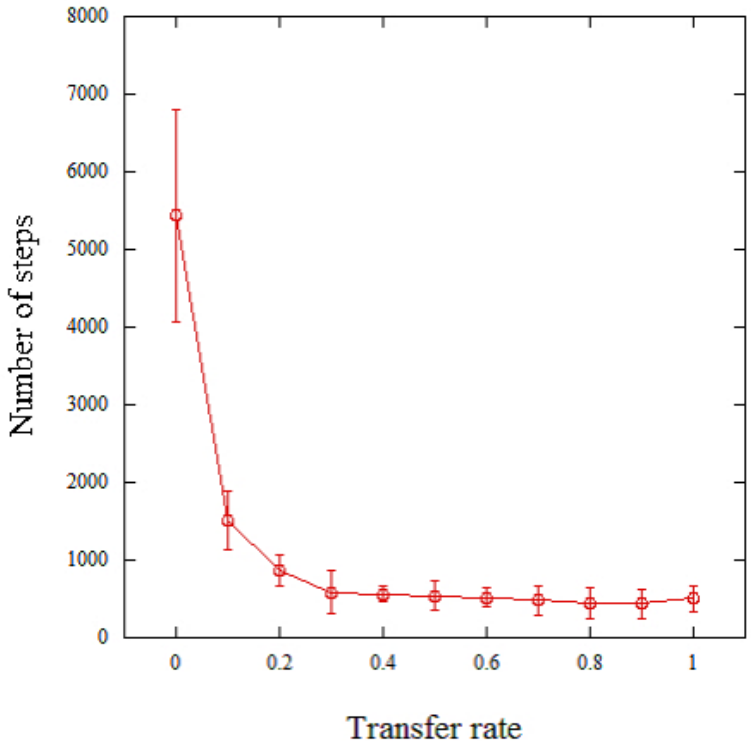


図 4.11 最短経路問題における転移曲線

ンスの良い知識の獲得を意味している．多少の増減や標準偏差のバラつきはあるが，転移率 = 0.2 より高いステップ数においては安定した収束値を得ている．

4.5.3　考察

4.5.3.1　動的環境の転移曲線に対する考察

動的環境において，転移率を上昇させるとジャンプスタートが発現することを確認した．これは，静的環境での結果と同様に知識の再利用の効果が発現していることを意味する．

図 4.11 において，転移率 = 0 のステップ数から転移率 = 0.1 のステップ数までの勾配が急であるのは，Target task である 3 vs. 1 タスクは 3 台のハンターが同時に獲物を捕獲する必要があるため，元々難易度が高く初期ステップ数が 4000step 以上と高くなっている．知識の転移を行うと，全てのハンターが他のハンターと協調して獲物を捕獲するための知識を少なからず持っているため，急激に初期ステップ数が低下すると考えられる．

実験結果から，本実験においても静的環境と同様にジャンプスタートを得たい場合は，転移率を高く設定することが良いことがわかる．

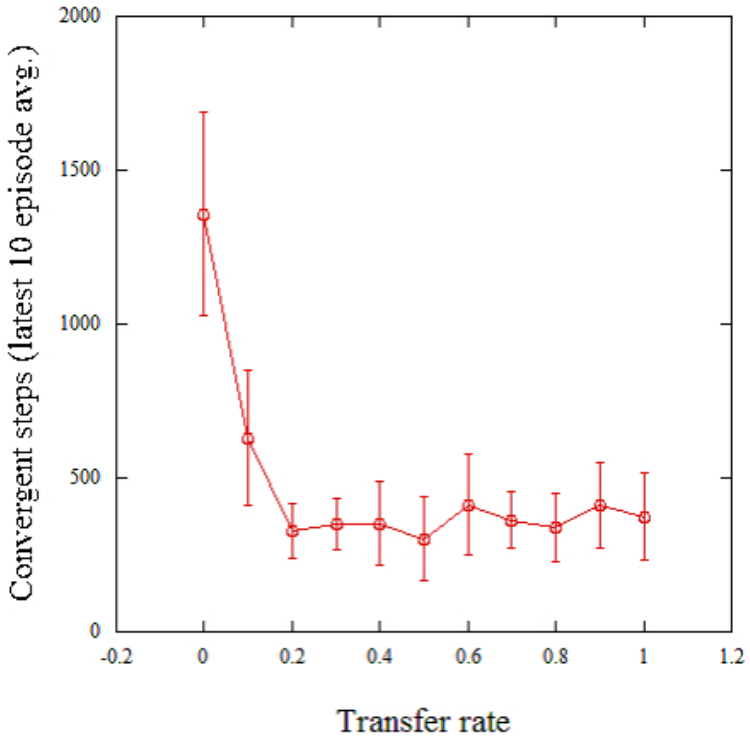


図 4.12 動的環境における収束値曲線

4.5.3.2　動的環境の収束値曲線に対する考察

転移率 = 0 の収束値に比べて，知識の転移を行ったエージェントの収束値は低い．これは，原因の一つとしてタスクの難易度に対する学習パラメータのチューニングが考えられる．強化学習は複数の学習パラメータを，環境に対して適切に設定しなければならない．経験的に良好な結果を発現させるパラメータは存在するが，学習をしてみなければわからないのが現実である．2 vs. 1 タスクでは有用であった学習パラメータは，3 vs. 1 タスクでも有効とは限らない．これにより，転移無（転移率 = 0）が本来到達可能なパフォーマンス（収束値）に到達できなかったと考えられる．

さらに，動的環境では静的環境の最短経路問題と異なり，獲物が移動し確率的にはハンターに近づいてくる行動もとる可能性がある．言い換えれば，静的環境のゴールが近づいてくるイメージである．もちろん，3 台のハンターで獲物を捕獲するため難易度は最短経路問題より高い．しかし，獲物の移動により偶然の捕獲が発生し，環境（Target task）を再学習する機会が増えると考えられる．これにより，予め簡単なタスク（Source task である 2 vs. 1 タスク）の知識を持ち，Target task を遂行することで，より最適に近いパフォーマンスが獲得できたと考えられる．

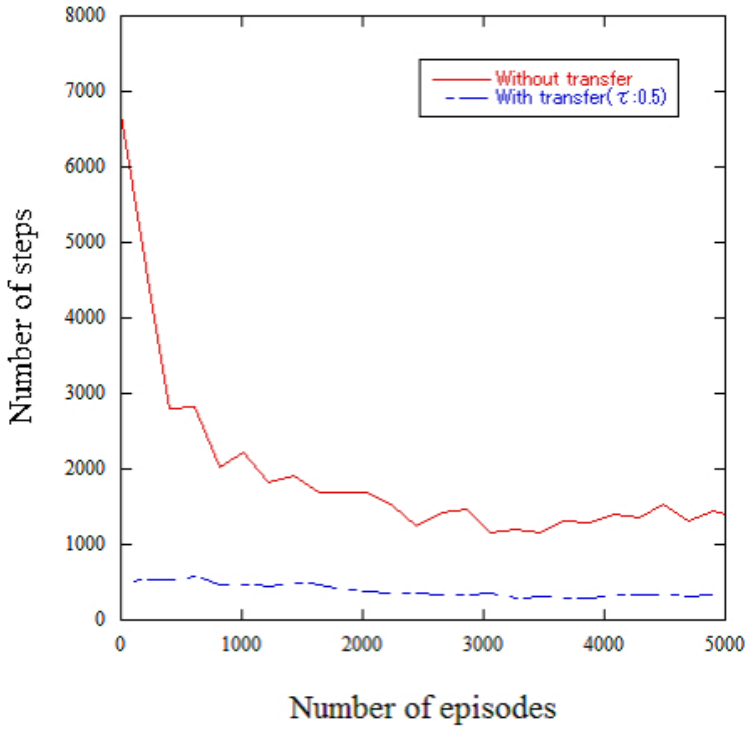


図 4.13 転移率 $\tau = 0.5$ の時の学習曲線

結果の例として，転移率 = 0.5 の時の学習曲線を図 4.13 に示す．この学習曲線は，付録 B に示すスムージング処理を行いグラフの描画を行っている．図 4.13 は，転移を行わないエージェントの学習曲線である “Without transfer” と，知識を転移したエージェントのエージェントの学習曲線 “With transfer” を示している．これらの学習曲線から，学習初期における明らかなジャンプスタートと，収束の傾向が表れた時点での収束値差が確認できる．また，“With transfer” の学習曲線は，緩やかであるがステップ数の低下傾向が表れている．これは，知識を再利用しても Target task にて再学習プロセスが発生していると考えられる．

結果としては，収束値という評価指標に対しても，動的環境では転移率は高い方が有用であることが示された．

4.6 結言

本章では，ヘテロジーニアスなマルチエージェント環境の転移学習において，効果的に知識の再利用を行うための知識結合法を新たに提案し，最短経路問題と追跡問題の計算機実験により議論した．これにより，以下の成果が得られた．

(1) 静的環境におけるヘテロジーニアスなエージェント間における転移学習では，転移率は $0.3 < \tau < 0.9$ の範囲で調整・設定することによりジャンプスタートが発現することを確認した．

(2) 動的環境におけるヘテロジーニアスなエージェント間の転移学習においても，$0.4 < \tau < 1$ の範囲で調整・設定が必要であることを確認した．

上記の成果より，Taylor らの転移学習手法では転移できない知識も，本章で提案した知識結合式により知識の再利用度合を調整することで，転移学習の効果を得ることが可能であることを示した．また，実験結果より転移率 τ は高い値の方が知識の再利用に対して効果的であることが示唆される．

第5章　強化学習における自律的収束推定法

5.1　緒言

本章では，エージェントが自律的に知識の共有や再利用を行うための，強化学習の収束推定法に関して議論を行う．表1.1に示した通り，本章は1.4節に提示した以下の 議論点に対してアプローチを行う議論の1つである．

- 知識獲得の処理に関する議論

これまで，環境に対する最適な振る舞いを獲得するモデルフリーの強化学習では，一般的に学習曲線や報酬曲線により学習の収束（知識の獲得）を判断する．しかし，この判断方法として，グラフとして描画された学習曲線や報酬曲線を，目視で直感的に人間（収束判定者と呼ぶこととする）が判断することが一般的である．また，学習が収束したとしても，基本的に強化学習という枠組みでは，学習を終了するプロセスは考慮されておらず，常に学習を行う．

知識の再利用法である転移学習を行う場合，学習途中の知識の転移は効果的であるとは考えにくい．また，目視による学習収束の判断は，転移学習を逐次手動で行わなければならないこをと意味し，そのプロセスは，著者らの提案した知識共創フレームワークと適合しない．さらに，数多くの不確定なエージェントが，知識の獲得・再利用を繰り返す場合，収束判定者の作業量を考えると現実的でない．したがって，以下の課題を設定する．

課題：「エージェント自身で知識の獲得を判断する手法の検討が必要」

本章では，学習曲線におけるフラクタル性を仮定し，強化学習エージェントがフラクタル次元解析を用いることで．自律的に学習の収束を推定可能にする手法を提案する．また，シングルエージェントとマルチエージェント環境に置いて，収束の推定が可能か，本手法を計算機実験により評価する．

5.2　学習曲線の収束

強化学習は，学習の結果として得られる学習曲線により学習の進度を見ることができる．学習曲線は，例えば図 5.1 のように横軸に学習の繰り返し回数，縦軸にエージェントのパフォーマンスを表すグラフにおける曲線である．学習曲線において，学習の繰り返しに伴いパフォーマンスが向上すると，学習が進み最適な方策に接近していると判断できる．環境が MDP では，学習曲線の収束が得られやすい．しかし，環境が non-MDP であれば，動的な環境であるため学習曲線が 1 つの値に収束しない．

学習曲線に振動が発生するが，初期パフォーマンスより明らかな改善がみられる場合，エージェントが準最適解を獲得したとみなせる．したがって，明らかに収束している学習曲線と，収束に近づいているが振動している学習曲線の状態を推定することが，知識の獲得を判断する手法となり得る．

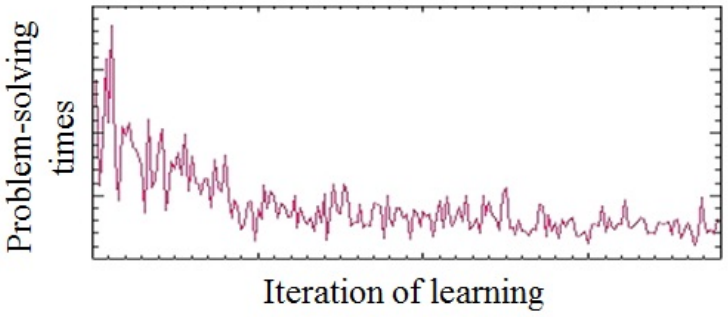


図 5.1 動的な環境における学習曲線の例

5.3　収束の判定法

強化学習に限らず，振動する波形の収束を推定する手法の研究がなされている．本節では，強化学習における収束推定の既存研究を俯瞰し，問題点を述べる．また，強化学習以外における収束の推定を行う研究を，関連研究として概説する．

5.3.1　強化学習における収束の推定

これまでの強化学習の収束判断において，収束推定者による目視が主であった．例えば，山口らや鯉江ら，Sichao らの研究においては，論文中に収束を目視で判断することを明示的に記している [78–80]．一方，簡易的な収束の判定法として，池上らの収束の判

断法があげられる [81]．池上らの収束の判断法は，強化学習における学習による行動価値の変化に着目した．池田らの手法は，行動価値の変化の傾きと閾値を用いて収束判定する手法である．閾値を用いる収束の判定は，直観的に理解しやすく自律的な収束推定機能としての実装も容易であると考えられる．しかし，学習環境や学習器，行動選択器などにより，異なる閾値の決定が必要であることは明らかであり，汎用的であるは言い難くシチュエーションごとに閾値の設定が必要であると考えられる．

他方，強化学習における収束を，収束判定者の主観や経験に依存しないで判定する研究も存在する．伊藤らは，学習残エントロピーを用いて Profit sharing の学習収束の判断を行っている [82]．学習残エントロピーは次式により定義される．

$$I(S) = -\frac{1}{\log 5} \sum_{a} p(a, S) \log p(a, S) \tag{5.1}$$

ある環境状態 S における，行動選択確率 $p(a, S)$ から式 (5.2) を定義している．文献 [82] において，伊藤らは “各状態において行動の不確定性がどの程度残っているかという指標である．学習残エントロピーは未学習状態では（行動に 5 つの選択肢があるとする）$\frac{\log 5}{\log 5} = 1$ であるが，学習が進行すると小さくなり，完全に特定の行為のみが選択されるようになると 0 になる．” と述べている．すなわち，学習により獲得された行動選択確率から，学習の進度を計算している．

また伊藤らは，学習残エントロピー $I(S)$ を平均した平均学習残エントロピーを次式のように定義している．

$$I = \frac{1}{|E|} \sum_{S_i \in E} I(S) \tag{5.2}$$

ここで，$|E|$ はエピソード E に含まれる状態数としている．伊藤らは閾値に相当する条件 “$I \leq 0.5$” を設定し，学習の収束などを判定している．上記手法の利点は，閾値以外の調整パラメータが無いことである．しかし，文献 [82] でも述べられている通り全ての学習に有効ではない．また，I を求めるときは，あるエピソードに含まれるすべての状態空間における行動価値を計算する必要がある．さらに，伊藤らの手法は知覚情報（エージェントのセンサ情報）の粗視化が行えるエージェントへの実装が前提となっており，効果の検証も含め，実装段階ではまだ汎用的な手法とは言い切れない．

5.3.2　強化学習以外における収束の推定手法

強化学習以外において，関数や数列の収束を推定する手法は多く存在する．古典的な収束の判定法として，ϵ-δ 論法 (ϵ-δ Limit Definition) やダランベールの収束推定法 (d'Alembert's test for convergence)，がある．しかし，これらは数学的な議論であり，極限や任意の数 ϵ を扱うと実世界での応用には厳格すぎる基準となる．なぜならば，環境毎における任意の数の調整は難しく，関数や数列（本研究においては学習）が極限に到達するほどの時間をかけられないことも少なくない．

大谷らは，マルチロボットシステムによる大規模構造物組立における，使用するロボットの台数とそれらの故障率を考慮した，組み立て成功率を評価している [83]．大谷らの研究では，ロボットの台数変化と故障率の違いによる組み立て成功率の最高値への収束，平均組み立て時間の最短時間へ収束を評価し，故障ロボットの回収という要素を導入したマルチロボットシステムの有用性を主張している．各故障率におけるロボットの使用台数変化に対して，組み立て成功率や組み立て時間の標準偏差が，それぞれの平均値の 5%以内であれば，組み立て成功率や組み立て時間が収束であると判定している．大谷らは，5%という値が工学的に一般的な値であると述べているが，強化学習の収束推定においては厳しい基準であると考えられる．

5.3.3　強化学習における収束推定の課題

これまでの収束推定手法は，各研究において有用性が示されているが，以下に示す現象においては十分に議論がなされていないと考えられる．

- 環境により学習曲線の収束値が異なる．
- 環境により収束時の学習曲線の振動が異なる．
- 学習の回数（学習曲線の定義域）は有限である．

学習曲線の収束推定では，単純な閾値などの基準は環境により収束値が異なるため，閾値設定が困難である．また，学習曲線は無限回の学習結果として得られないため，限られたデータ数で収束を推定しなければならない．

そこで，学習曲線の数値を基に収束推定するのではなく，本研究では学習曲線の形状に着目した．本章では，学習曲線のフラクタル性に着目し，学習曲線から切り出した部分曲線の形状のフラクタル次元を計算し，その推移により収束を推定する手法を提案する．もちろん全ての環境，条件に適応できる汎用的な収束推定法の確立は困難である．本研究

で提案する手法は，収束推定に用いていたパラメータを削減し少しでも汎用性も持たせた手法の確立が目的である．次節から，提案手法の基礎としてフラクタルに関して記す．

5.4　フラクタル次元解析

5.4.1　フラクタルとフラクタル次元

フラクタルは，Mandelbrot により提唱された幾何学における自己相似性を持った図形や曲線である [84]．フラクタルな特性を持つ図形や曲線は，それを無限に拡大しても同様の構造を有するものである．自然界には完全にフラクタルな物質や構造は存在しないことに注意が必要であるが，文献 [84] で計測された海岸線のフラクタル性をはじめ，フラクタルな特徴を有する物質や構造が，世界に多く存在することが明らかになっている．例えば，樹木の幹から葉まで伸びる形や [85]，岩盤などがある [86]．

上記以外にも，完全では無いが確率的・統計的自己相似である形状や特性も多く，観測対象の類似性評価や分類などの指標として，フラクタル性を定量的に表現する指標であるフラクタル次元の応用が成されている [87–90]．フラクタル次元は，実数で表現され例えば 1.23456 次元などと表現することができる．さらに，一般的に使われている次元数の 1 次元 (直線)，2 次元 (平面)，3 次元 (立体) とも整合性が取れる．すなわち，直線のフラクタル次元 D_F は 1 次元，曲線は $1 \leq D_F < 2$，平面は 2 次元，曲面は $2 \leq D_F < 3$ などのように直感的にとらえることが出来る [91]．代表的なフラクタルである，コッホ切片やシェルピンスキーの三角形などは 1 次元から 2 次元の間でフラクタル次元をとる．

5.4.2　フラクタル次元解析

フラクタル次元を算出する方法として，フラクタル次元解析がある．フラクタル次元の定義は様々なため，各フラクタル次元を算出するために，対応したフラクタル次元算出法（フラクタル次元解析）が存在する．本研究では，計算機実装が容易であり，メカニズムの直感的な理解が容易であるボックスカウンティング次元を用いる [92]．

ボックスカウンティング次元は以下の手順により算出が可能である．まず，測定対象である図 5.3(a) を，図 5.3(b) のように任意のサイズ δ のグリッドでマスクする．次に，これを様々なサイズの δ で測定し，図 5.2 のように近似した直線の傾きがフラクタル次元となる．また，近似直線の傾きの算出は次式で定義される．

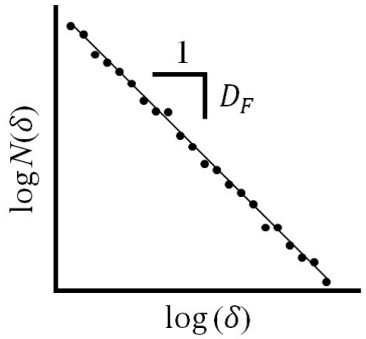


図 5.2 任意の δ によるフラクタル次元 D_F の算出

$$D_F = -\lim_{\delta \to 0} \frac{\log N(\delta)}{\log(\delta)} \tag{5.3}$$

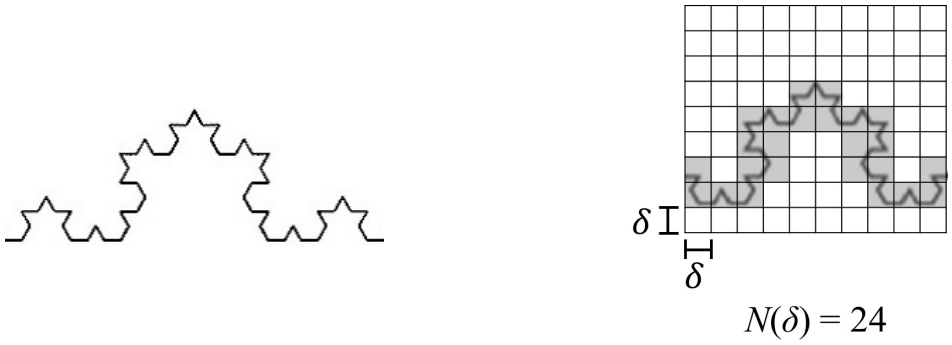
通常，フラクタル次元解析は Image J や Fractal dimension estimator などのフラクタル次元解析の機能が実装された画像処理ソフトを用いる．本研究では，フラクタル次元解析にアメリカ国立衛生研究所 (National Institutes of Health: NIH) から配布されている Image J を用いる [93].

ここで，注意が必要なのは，式 (5.3) を用いるために，理論的には観測対象が次式に示す関係を満たしていることである [94, 95].

$$N(\delta) \sim c\delta^{-D_F} \tag{5.4}$$

ここで，c は任意の倍数であり，δ に対するスケール因子と呼ばれる．既に述べたが，実際に測定する図形や構造は数学的な対象物とは異なり，完全な自己相似性を示さない [94]. しかし，式 (5.4) を完全に満たさなかったとしても，測定対象がフラクタル性を有している可能性を評価することはできると考えられる.

式 (5.4) を確認するために，ベキ分布との相関係数により評価できると考えられる．自己相似性は，その特徴量がベキ分布に従う [94]．図 5.2 における，両対数上の直線近似はベキ分布との近似である．したがって，フラクタル次元解析における直線近似との相関係数 R^2 が高ければ，ベキ分布との相関性も高く，自己相似性を有していると考えられる [96].



(a) コッホ曲線　　(b) コッホ曲線のボックス数

図 5.3 ボックスカウントの例

5.5　フラクタル次元解析を用いた収束推定

前節まで説明したフラクタル次元解析を応用し，エージェントが自律的に強化学習の収束を推定する手法を本節で述べる．また，提案手法を Convergence estimation on franctal dimensional analysis (CEFD) と呼ぶこととする．

また，次項にて提案する手法は，フラクタル次元解析を用いた収束推定法の有用性を確認するために，エージェントの内部プログラムとしての実装は考慮しない．エージェントが学習を終了し，その学習の結果として得られる学習曲線を提案手法を用いて評価する．エージェントへの実装を考慮した手法は 5.5.2 項にて述べる．

本研究では，画像処理を用いた CEFD を CEFDi (CEFD with image processing) と呼び，実装を考慮した数値計算による CEFD を CEFDn (CEFD with numerical processing) と呼ぶこととする．

5.5.1　CEFDi

画像処理を用いた CEFD である CEFDi の処理概略を図 5.4 に示し，以下に処理の流れを概説する．

(1) まずエージェントが学習により得られた学習曲線を描画し，任意のエピソード数（横軸）とステップ数（縦軸）に分割する．この時，分割された全ての学習曲線の値域は学習初期の最大値に設定される．

(2) 分割された学習曲線をそれぞれ，正方形画像として保存する．そして，各画像をフラクタル次元解析ソフトウェアに入力し，フラクタル次元 D_F を算出する．

(3) 算出されたフラクタル次元をエピソード対フラクタル次元のデータ列として再構成し，フラクタル次元の推移を観測する．

(4) フラクタル次元が環境に併せた閾次元を下回る学習曲線は準収束とし，上回る場合は非収束と推定する・

図 5.4 は，フラクタル次元解析可能な画像処理ソフトを利用することを前提とする．フラクタル次元解析のための画像処理ソフトには，アメリカ国立衛生研究所 (NIH) から配布されている Image J とプラグインである FracLac を用いる．Image J によるフラクタル次元解析は，文献 [92] などでも用いられており実績がある．

ここで注意が必要なのは，上述の処理はあくまでも基礎的検討として，学習が終了した学習曲線のフラクタル次元を計測している点である．学習エージェントへの実装を前提とする場合は，フラクタル次元解析処理を学習中に実行させ，学習曲線のフラクタル次元を評価する必要がある．エージェントへの実装を考慮した CEFD は 5.5.2 節にて説明する．また，学習曲線を描画・分割するために，ある程度の学習回数，少なくても数百エピソード数以上の学習を繰り返す必要があることに注意が必要である．

5.5.2 CEFDn

前項にて提案した CEFDi では，学習結果として得られる学習曲線を画像処理しているため，計算負荷や処理速度などの問題から強化学習エージェントやロボットへの実装が課題となる．そこで，ここでは学習と同時にステップ数からシームレスにフラクタル次元解析を行い，収束を推定する手法を提案する．これにより，フラクタル次元解析するための一般的な画像処理アプリケーションを，エージェントが自動的に実行可能な形式に改良しなくても，学習過程から得られるステップ数から直接フラクタル次元解析が可能となる．

5.5.2.1 フラクタル次元の計算法

画像処理を用いるフラクタル次元解析以外にも，得られる数値情報から直接フラクタル次元を計算する手法が検討されている．例えば，山際の金属破面をフラクタル次元により評価する手法がある [91,97]．山際は，測定対象のボックス数 n_i は次式により容易に

評価できると述べている．

$$n_i = \frac{H_{max} - H_{min}}{h} \tag{5.5}$$

ここでボックス数 n_i は，さらに次式を用いて総ボックス数をカウントする．

$$N = \sum_{i=0}^{m} n_i \tag{5.6}$$

上式における N は，式 (5.3) の $N(\delta)$ と同義である．上記はあくまでもフラクタル次元解析の近似法であるにすぎないことに注意が必要である．

山際の手法は，金属破面という連続状態の情報から，画像処理による離散化（粗視化）した情報をフラクタル次元解析するために，式 (5.5) を提案してる．金属破面の情報は，カメラ画像であるため離散化された状態ではあるが連続に近い．本提案手法における学習曲線の値は高々数万，数千の正の整数値であり，さらなる計算法の単純化・高速化に対して検討の余地があると考える．

山際のフラクタル次元解析法を基に，本研究では新たに数値処理を用いた CEFD である CEFDn を提案する．CEFDn におけるフラクタル次元解析の計算式は次式で定義する．

$$n_i(\delta) = \left\lfloor \frac{S_i^{max}}{\delta} \right\rfloor - \left\lfloor \frac{S_i^{min}}{\delta} \right\rfloor + 1 \tag{5.7}$$

$$N(\delta) = \sum_{i=1}^{e/\delta} n_i(\delta) \tag{5.8}$$

ここで，δ はボックスサイズを意味し，S_i^{max} はある学習曲線の測定区間 e の i 番目 ($1 \leq i \leq \frac{e}{\delta}$) における最大ステップ数であり，$S_i^{min}$ は最少ステップ数である．どちらも S_i^{max}, $S_i^{min} \in \mathbb{N}^+$ である．また，$n_i(\delta)$ は測定区間 e 内の i 番目における，ボックス δ が学習曲線を覆うのに必要なボックス数を意味する．すなわち，$N(\delta)$ は学習曲線の測定区間 e をボックスサイズ δ で測定した時におけるボックス総数である．$\lfloor \cdot \rfloor$ は小数部分を切り捨てる床函数である．

式 (5.8) の計算例を図 5.5 に示す．例えば 10 個の学習曲線データ（図 5.5 の赤線）が強化学習より得られたとする（実際の測定区間エピソード e は 100 episode など用いる）．ボックスサイズ δ を 4 とした時，最初に n_1 を計算する．エピソードの数と，ステップ数

の数はそれぞれ δ と対応し，ボックス数 n_1 は

$$\begin{aligned} n_1 &= \left\lfloor \frac{9}{4} \right\rfloor - \left\lfloor \frac{4}{4} \right\rfloor + 1 \\ &= \left\lfloor 2.25 \right\rfloor - \left\lfloor 1 \right\rfloor + 1 \\ &= 2 \end{aligned} \tag{5.9}$$

となる．さらに，ボックス数 n_2 は

$$\begin{aligned} n_2 &= \left\lfloor \frac{3}{4} \right\rfloor - \left\lfloor \frac{1}{4} \right\rfloor + 1 \\ &= \left\lfloor 0.75 \right\rfloor - \left\lfloor 0.25 \right\rfloor + 1 \\ &= 1 \end{aligned} \tag{5.10}$$

このカウントしたボックスの数 n_i を e/δ まで計算し，それらの総和をとれば $N(\delta)$ となる．

また，CEFDn のエピソード全体の処理概念図を図 5.6 に示す．図 5.6 において，Sweep はフラクタル次元解析を行う間隔であり，本論文では Sweep 間隔を p とする．CEFDi では基礎的な検討であったため，学習曲線を任意の間隔で分割しフラクタル次元解析を行っていた．CEFDn ではステップ数が常に得られるため，測定区間エピソード e の分だけステップ数が得られれば，それ以降は最短で 1 episode 毎にフラクタル次元解析が実行可能である．

5.5.2.2　エージェントへの実装法

本節にて提案する手法は，画像処理による CEFD と異なり強化学習エージェントが学習を行いながら収束を推定する．そのため，プログラムで実装可能なアルゴリズムとして構築する必要がある．実装のための疑似コードを，強化学習アルゴリズムと共にアルゴリズム 1 に示す．

Algorithm 1 数値処理を用いたフラクタル次元解析アルゴリズム

Initialize $Q(s,a)$ arbitrarily
$\delta_i \in \{1, 2, \cdots, n\}$
Initialize N arbitrarily number
Initialize T_h arbitrarily threshold value of fractal dimension
Initialize $COUNT$
Initialize D_f and D'_f
while $(D_f < T_h) \cap (COUNT > N)$ **do**
  while Capture state $\neq$ true **do**
    Observe state s
    $a \leftarrow \frac{\exp\left(\frac{Q(s,a)}{T}\right)}{\sum_{b \in A} \exp\left(\frac{Q(s,b)}{T}\right)}$
    Take action a
    Observe r and s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)\}$
  end while
  if Number of episode $> \max(\delta_i)$ such as n **then**
    for all n_i **do**
      $n_i(\delta) = \left\lfloor \frac{S_i^{max}}{\delta} \right\rfloor - \left\lfloor \frac{S_i^{min}}{\delta} \right\rfloor + 1$
      $N(\delta) = N(\delta) + n_i(\delta)$
    end for
    $D_f \leftarrow -\lim_{\delta \to 0} \frac{\log N(\delta)}{\log(\delta)}$
    if $D_f \geq D'_f$ **then**
      $COUNT++$
    else
      $D'_f \leftarrow D_f$
      $COUNT \leftarrow 0$
    end if
  end if
end while

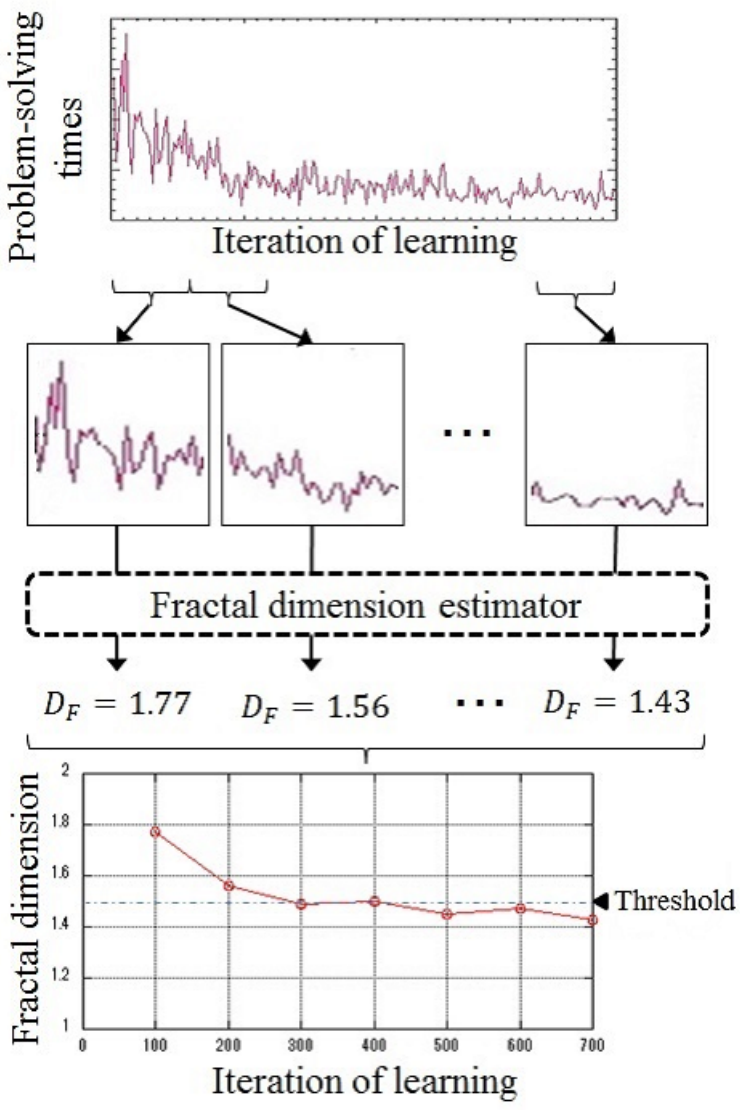


図 5.4 CEFDi の処理概略図

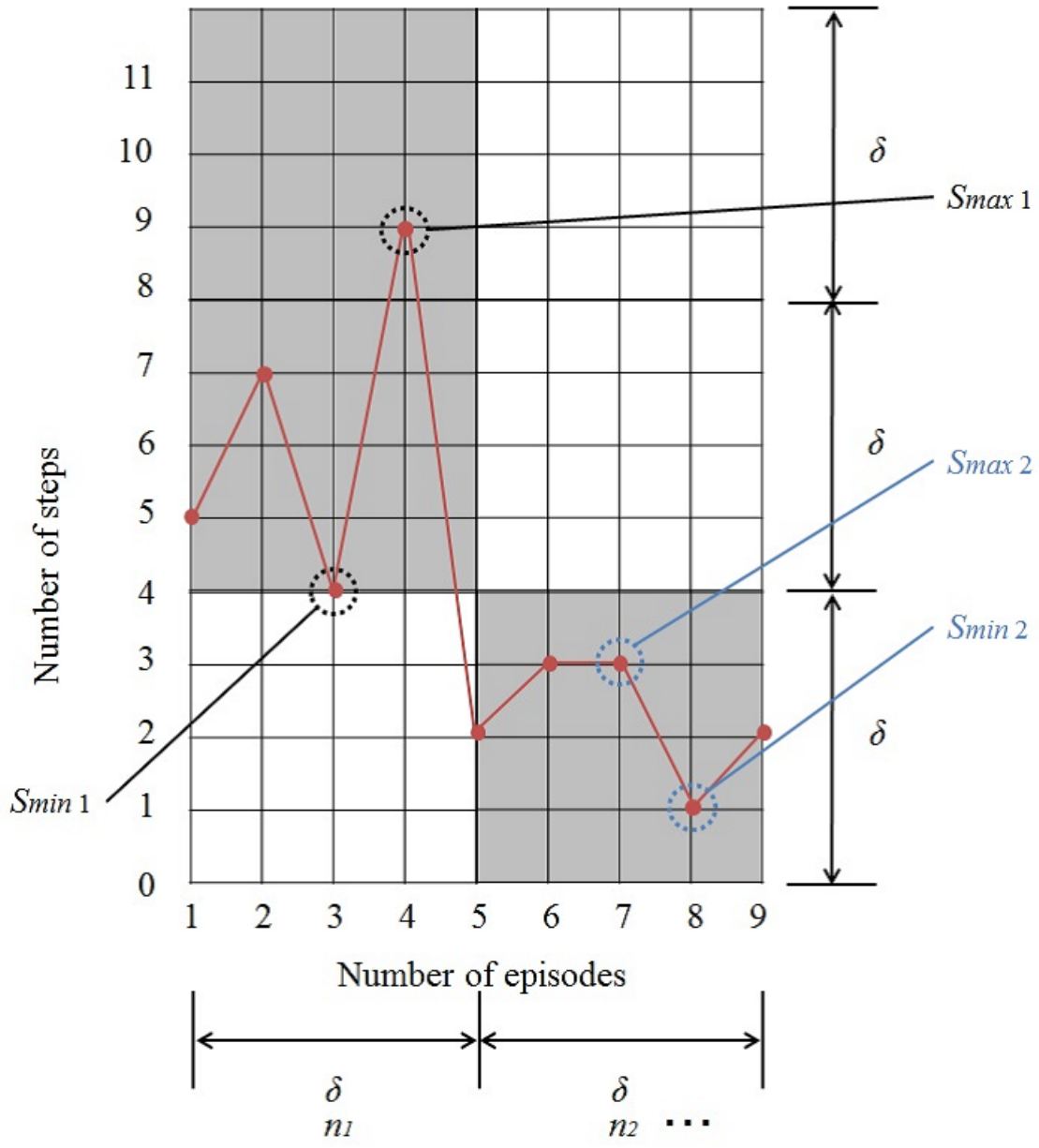


図 5.5 CEFDn の計算例

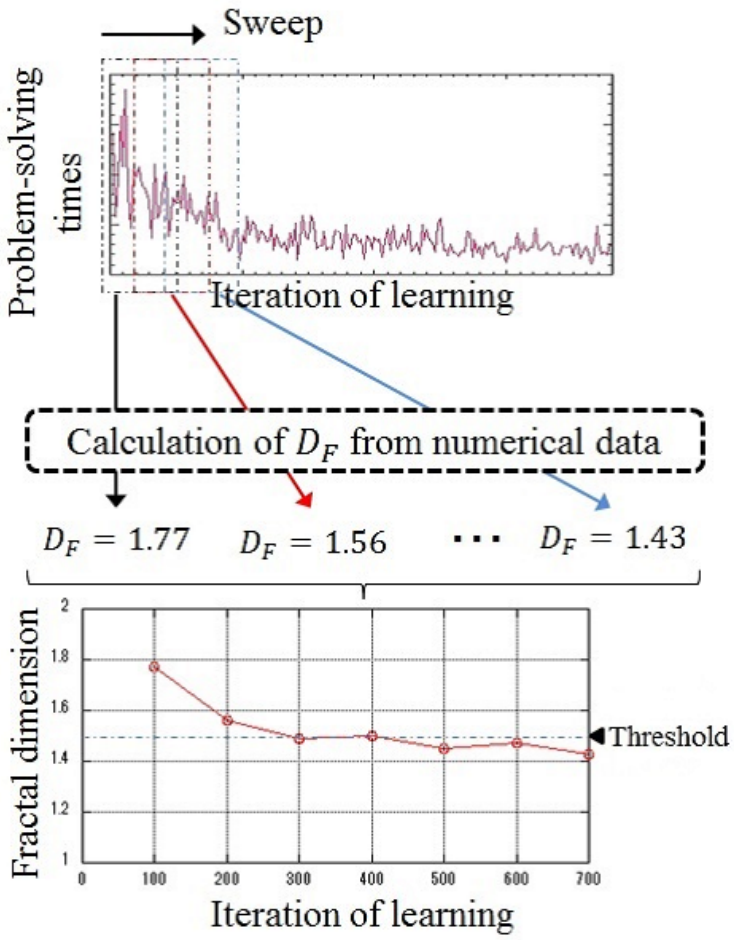


図 5.6 CEFDn の処理概略図

5.6 CEFDi の計算機実験

本実験では，画像処理を用いた CEFD を，計算機実験により評価する．強化学習エージェントのタスクとしては，これまで述べた追跡問題を採用し，静的環境や動的環境を用いて画像処理を用いた CEFD の有用性を評価する．

前述の通り，動的環境においては収束の定義ができない，また強化学習により学習エージェントのパフォーマンスが向上しているが明らかに収束しているとは言い難い状態が存在する．本実験では，提案手法によりタスク別の閾値を設定することで学習エージェントの知識獲得を推定できるか検証する．

5.6.1 問題設定

本実験においても，ハンターが獲物の捕獲行動を学習する追跡問題を採用する．これまでと同様に，ハンターが獲物を捕獲するまでの行動回数（ステップ数）を学習曲線として結果を出力し，その学習曲線を CEFDi により収束の推定が可能であるか評価する．

5.6.1.1 環境設定

本実験では，環境の違いに対する CEFD の有用性も評価するために，4 種類の広さのグリッドワールドにて実験を行う．グリッドワールドは図 5.7 を用いた．各グリッドワールドにおけるエージェントの初期位置を表 5.1 に記す．

5.6.1.2 エージェント設定

本実験のエージェントは図 5.8 に示す，ハンターと獲物を用いる．複数台のハンターを用いる実験においては，図 5.8(a) のハンターを 2 台用いることとする．ハンターは図 5.8(a) のように，灰色でマスクされた周囲 5 × 5 の視覚範囲を有しており，獲物は図 5.8(b) のように縦横 4 セルの視覚範囲を有する．獲物は基本的にランダムに行動を選択し，視覚範囲内にハンターが存在する場合のみ逃避行動をとることが可能である．

収束を推定する強化学習エージェントはハンターとし，獲物は学習しない．ハンターの強化学習器は，これまでと同様に Q 学習（式 (2.1)）を用いる．捕獲条件は，これまでの追跡問題と同様に，全てのハンターが獲物と隣り合う座標に到達したときとする．各エージェントは初期座標から移動を開始し，獲物を捕獲した後に初期座標にリセットされる．各エージェントの行動順序はハンター 1→ ハンター 2→ 獲物の順とし，ハンター 2

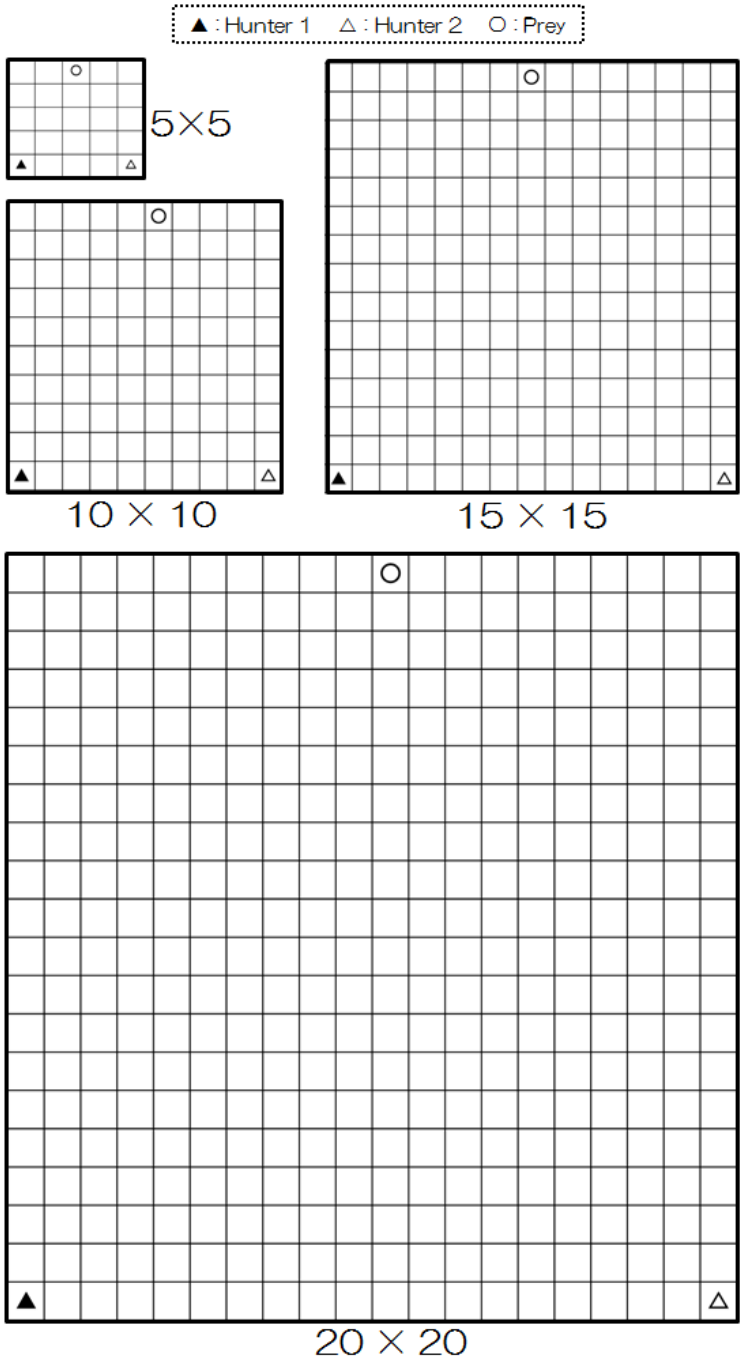
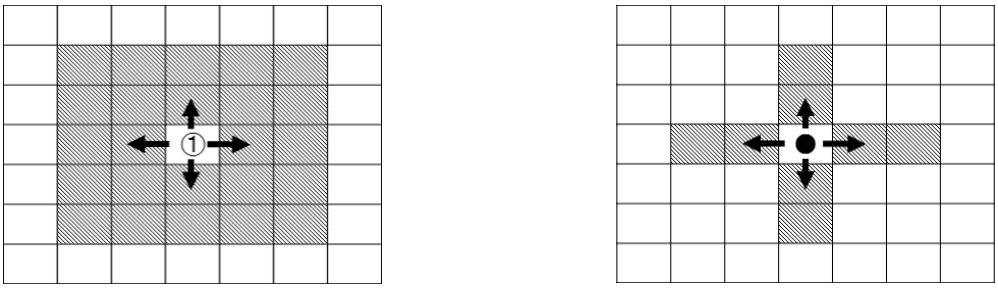


図 5.7 4 種類の広さのグリッドワールドとエージェントの初期位置例

表 5.1 Initial position in each agents

Condition	Field size	Hunter1	Hunter2	Prey
Condition a (Static)	5 × 5	(1, 1)	-	(5, 5)
	10 × 10	(1, 1)	-	(10, 10)
	15 × 15	(1, 1)	-	(15, 15)
	20 × 20	(1, 1)	-	(20, 20)
Condition b (Dynamic)	5 × 5	(1, 1)	-	(5, 5)
	10 × 10	(1, 1)	-	(10, 10)
	15 × 15	(1, 1)	-	(15, 15)
	20 × 20	(1, 1)	-	(20, 20)
Condition c (Multi-hunter)	5 × 5	(1, 1)	(5, 1)	(3, 5)
	10 × 10	(1, 1)	(10,1)	(6, 10)
	15 × 15	(1, 1)	(15, 1)	(8, 15)
	20 × 20	(1, 1)	(20, 1)	(11, 20)
Condition d (Non convergence)	5 × 5	(1, 1)	-	(5, 5)
	10 × 10	(1, 1)	-	(10, 10)
	15 × 15	(1, 1)	-	(15, 15)
	20 × 15	(1, 1)	-	(20, 20)



(a) ハンター　　(b) 獲物

図 5.8 CEFDi の計算機実験で使用するエージェント

を用いない実験条件においては，ハンター 2 の行動順番はスキップされる．全てのエージェントが 1 回行動した状態を 1 ステップと呼ぶ．

5.6.1.3　実験条件

本実験では，前述の追跡問題を基本に，次に示す 4 条件にて実験を行い，収束推定の環境依存性と閾値の検討，収束推定の可能性を評価する．

(1)　静的環境
(2)　動的環境（ハンター 1 台 vs. 獲物 1 台）
(3)　マルチハンター環境（ハンター 2 台 vs. 獲物 1 台）
(4)　非学習環境

(1) 静的環境は，学習可能で移動可能なハンター 1 台と，移動できない獲物 1 台の環境とし，環境の状態遷移がハンターの行動のみに依存する MDP 環境である．また，獲物は移動しないことから最短経路問題と同等のタスク設定である．(2) 動的環境は，ハンターが 1 台，移動可能な獲物 1 台の環境であり，環境の状態遷移がハンターの行動のみに依存しない non-MDP 環境である．また，追跡問題である．(3) マルチハンター環境は追跡問題で，non-MDP 環境であるが，(2) の動的環境よりさらに環境が動的な環境である．ハンターを 2 台，移動可能な獲物を 1 台用いる．また，捕獲条件として 2 台のハンターが同時に獲物と隣り合う座標に移動する必要があるため，タスクの難易度も高い．

以上までの実験条件は，様々な学習曲線の収束推定を評価するために行う実験である．(4) 非学習環境は，収束しない学習曲線を CEFDi で評価し，収束していないことを推定するための実験条件である．

本実験で用いる強化学習パラメータを表 5.2 に示し，CEFDi に用いる Image J のパラメータセッティングを表 5.3 に示す．

表 5.2 CEFDi の計算機実験におけるパラメータ設定

Parameter	Value
Learning rate	0.1
Discount rate	0.99
Reward	1
Boltzmann parameter	
Static	0.01
Dynamic and multi-hunter	0.005
Non-convergence	1.0
Default Q-value	0
Number of episode	10000
Number of trial	10

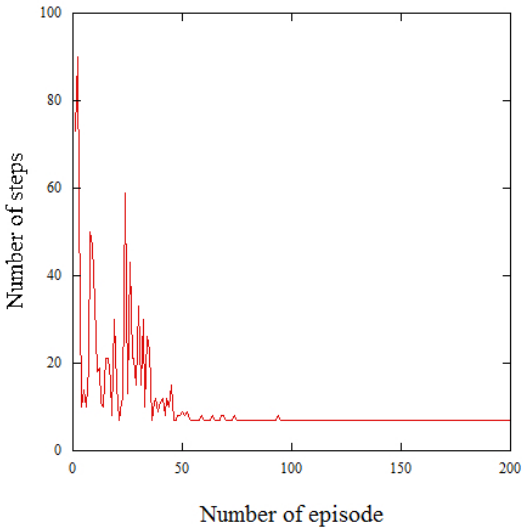
表 5.3 CEFDi に用いる Image J のパラメータ設定

Parameter	Value
Size of image	420 × 420 pixel
Binary or grayscale	“Autoconvert to binary”
Set scan back ground	“Let the program shoose”
Positions	1
Number of sizes	50
Minimum box size	2 pixel
Maximum bos size	50 pixel

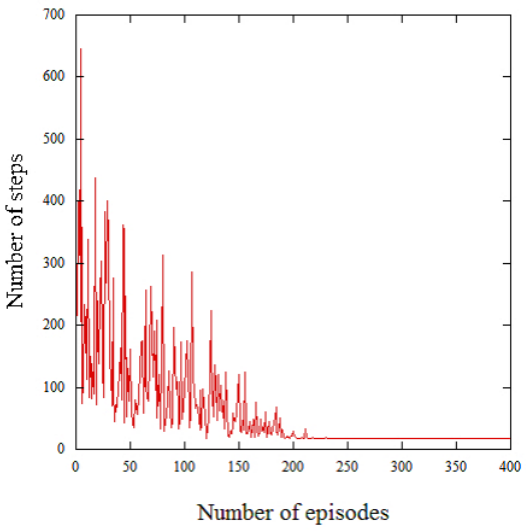
5.6.2 実験結果

5.6.2.1 静的環境における結果

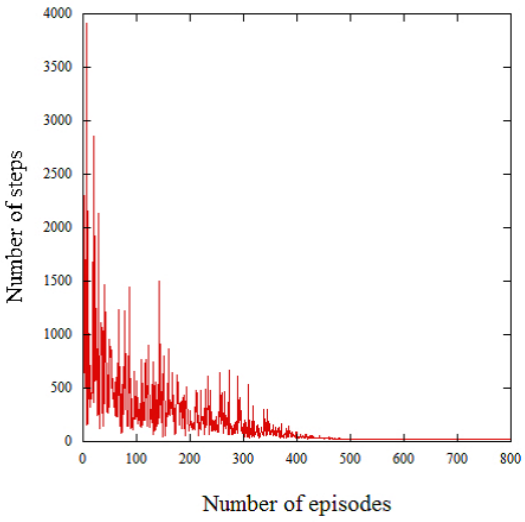
静的環境において，各グリッドワールドサイズの学習曲線を 1 つずつ図 5.9(a) から (d) に示す．図 5.9 をはじめ，本章で示す学習曲線は全て，平均やスムージング処理は行っていない，シミュレータから得られたオリジナルのデータを描画した学習曲線である．また，それらの学習曲線の CEFDi によりフラクタル次元解析した結果を図 5.10 に示す．各グリッドワールドサイズにおいて，10 回学習を行っているため各学習曲線をフラクタル次元解析し，フラクタル次元の推移の平均と標準偏差を図 5.10 に示している． いずれのグリッドワールドサイズにおいても，学習曲線（図 5.9(a)∼(d)）は高いステップ数から徐々に減少し，それぞれ 7step，17step，27step，37step に収束した．これらの収束値は，



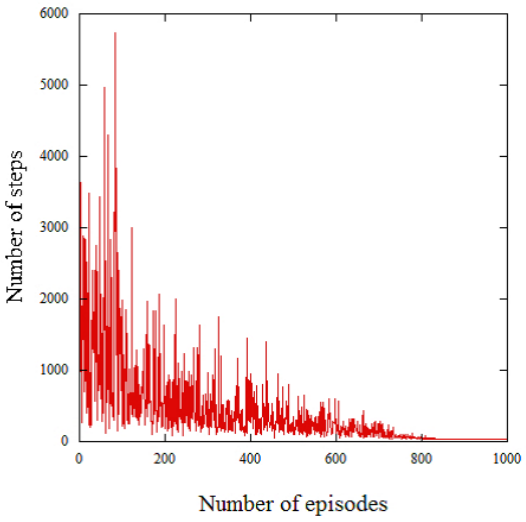
(a) 5 × 5



(b) 10 × 10



(c) 15 × 15



(d) 20 × 20

図 5.9 静的環境における各グリッドワールドの学習曲線（Trial 1）

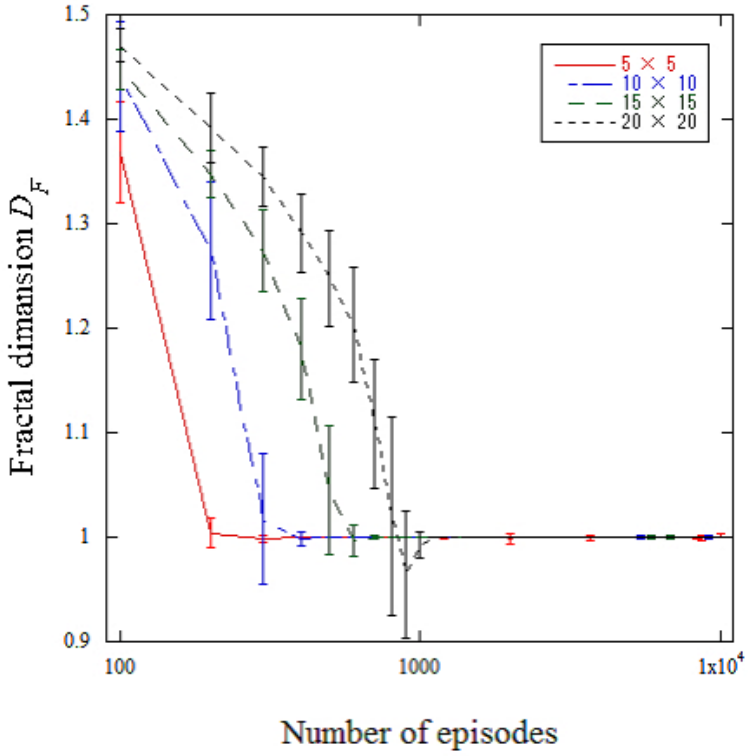


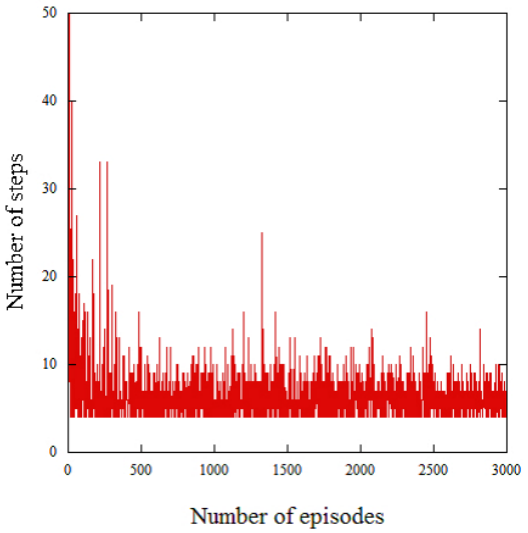
図 5.10 静的環境におけるフラクタル次元解析結果

最適解（ここでは最短経路）を学習していることを確認した．また，フラクタル次元としては 1.4 次元付近からフラクタル次元が低下し，学習曲線の収束時にはフラクタル次元が 1 次元と同様のフラクタル次元に収束する結果が得られた．収束時のフラクタル次元はいくつか偏差が見られる所があるが，1000 エピソード以降は 1 次元に収束していると言える．収束時の学習曲線を見ると，最短経路へ収束しているため学習曲線は直線となる．直線のフラクタル次元は 1 次元であることが知られているため，本実験条件では妥当な結果が得られたと考えられる．

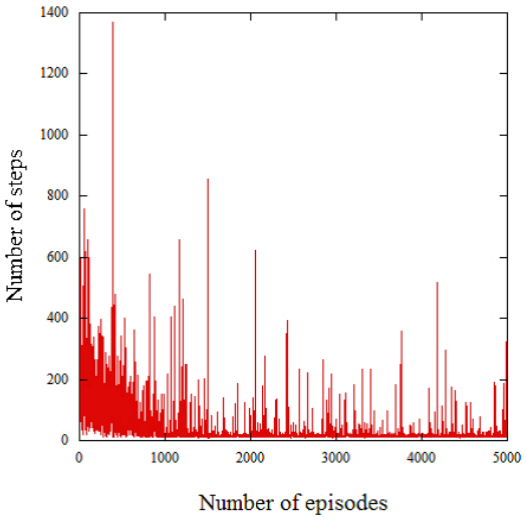
5.6.2.2　動的環境における結果

動的環境において，各グリッドワールドサイズの学習曲線を 1 つずつ図 5.11 (a) から (d) に示す．また，それらの学習曲線の CEFDi によりフラクタル次元解析した結果を図 5.12(a) から (d) に示す．

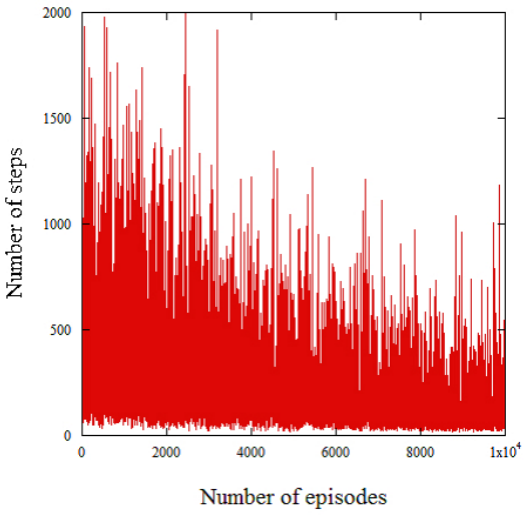
本実験の学習曲線は，グリッドワールドサイズ 5 × 5 と 10 × 10 に関して学習初期の高いステップ数から低いステップ数への収束傾向が現れている．これに対し，グリッドワールドサイズ 15 × 15 と 20 × 20 は学習曲線ではステップ数の低下はみられるが，収束する



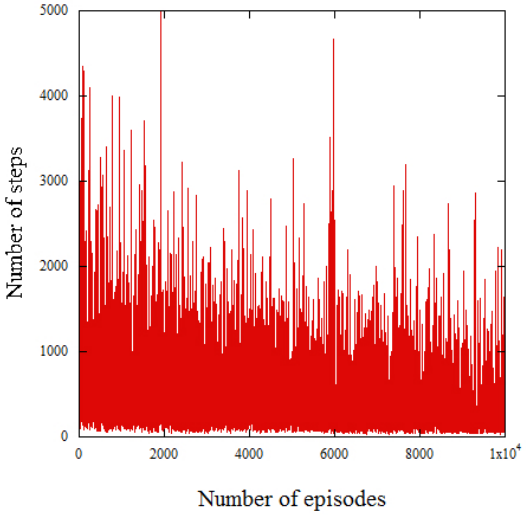
(a) 5 × 5



(b) 10 × 10

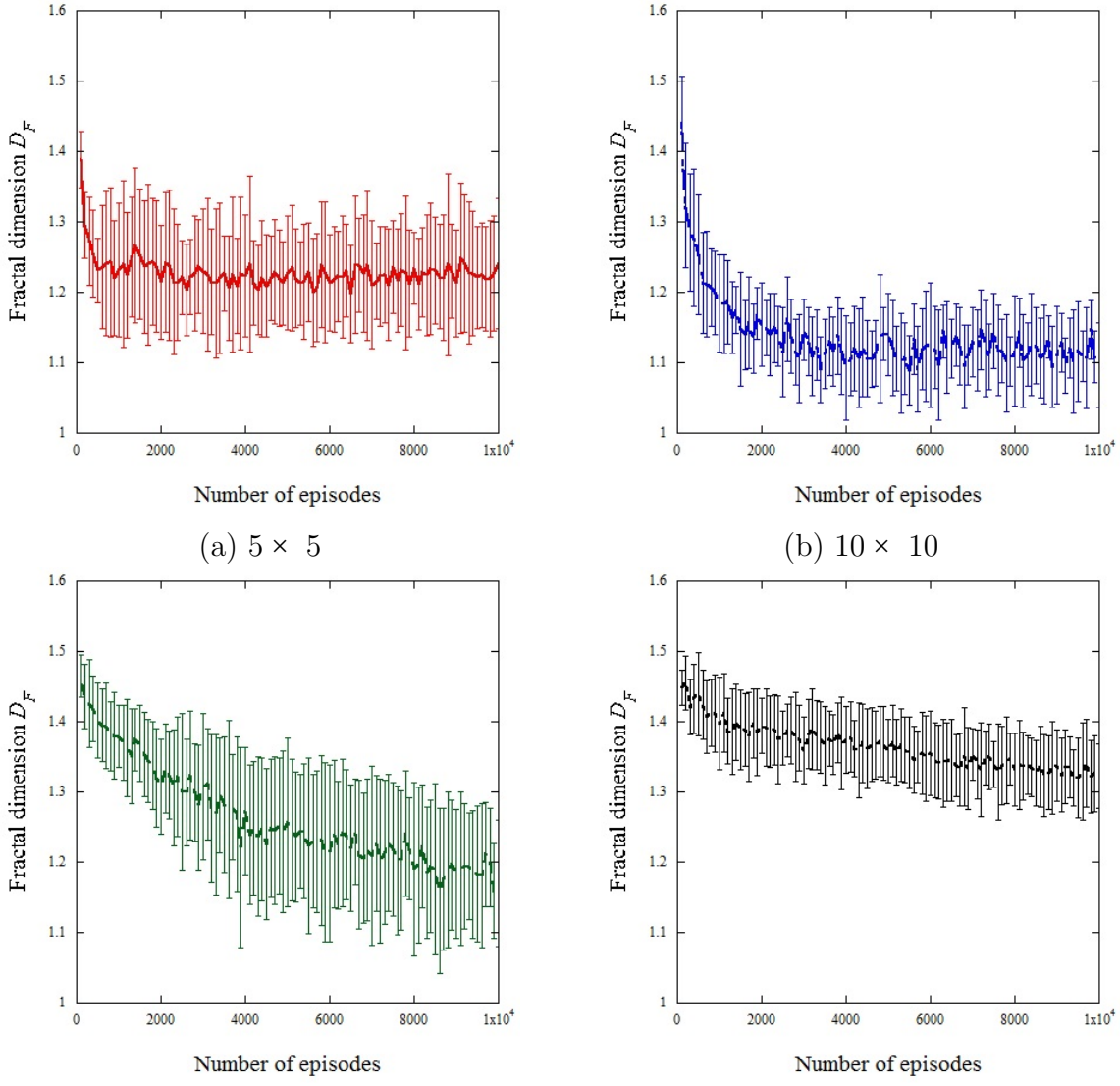


(c) 15 × 15



(d) 20 × 20

図 5.11 動的環境における各グリッドワールドの学習曲線（Trial 1）



(a) 5 × 5　　(b) 10 × 10

(c) 15 × 15　　(d) 20 × 20

図 5.12 動的環境におけるフラクタル次元解析結果

際のステップ数の安定は見えず，収束しているとは言い難い．

フラクタル次元の推移（図 5.12）から，グリッドワールドサイズ 5 × 5 と 10 × 10 はフラクタル次元の低下と，安定が現れている．これらの結果から，収束傾向が現れた学習曲線のフラクタル次元は 1.3 次元以下となっている．また，グリッドワールドサイズ 15 × 15 と 20 × 20 のフラクタル次元の推移は，学習曲線と同様に次元数が低くなっているが次元数の安定が現れていない．しかし，次元数の低下という結果より収束傾向は現れていると考えられる．

5.6.2.3　マルチハンター環境における結果

マルチハンター環境において，各グリッドワールドサイズの学習曲線を 1 つずつ図 5.13 (a) から (d) に示す．また，それらの学習曲線の CEFDi によりフラクタル次元解析した結果を図 5.14(a) から (d) に示す．

マルチハンター環境の学習曲線において，学習の初期段階におけるタスク達成が困難であるため，十分に学習をした状態のステップ数との差が大きい．図 5.13(a)∼(d) を見てもわかるとおり，学習を繰り返すごとに学習の初期ステップ数が低下し，収束の傾向が現れている．収束傾向後の学習曲線は，ステップ数の違いや分散の違いはあるが，学習の初期ステップ数から比較すると十分に準最適解が獲得できていると考えられる．

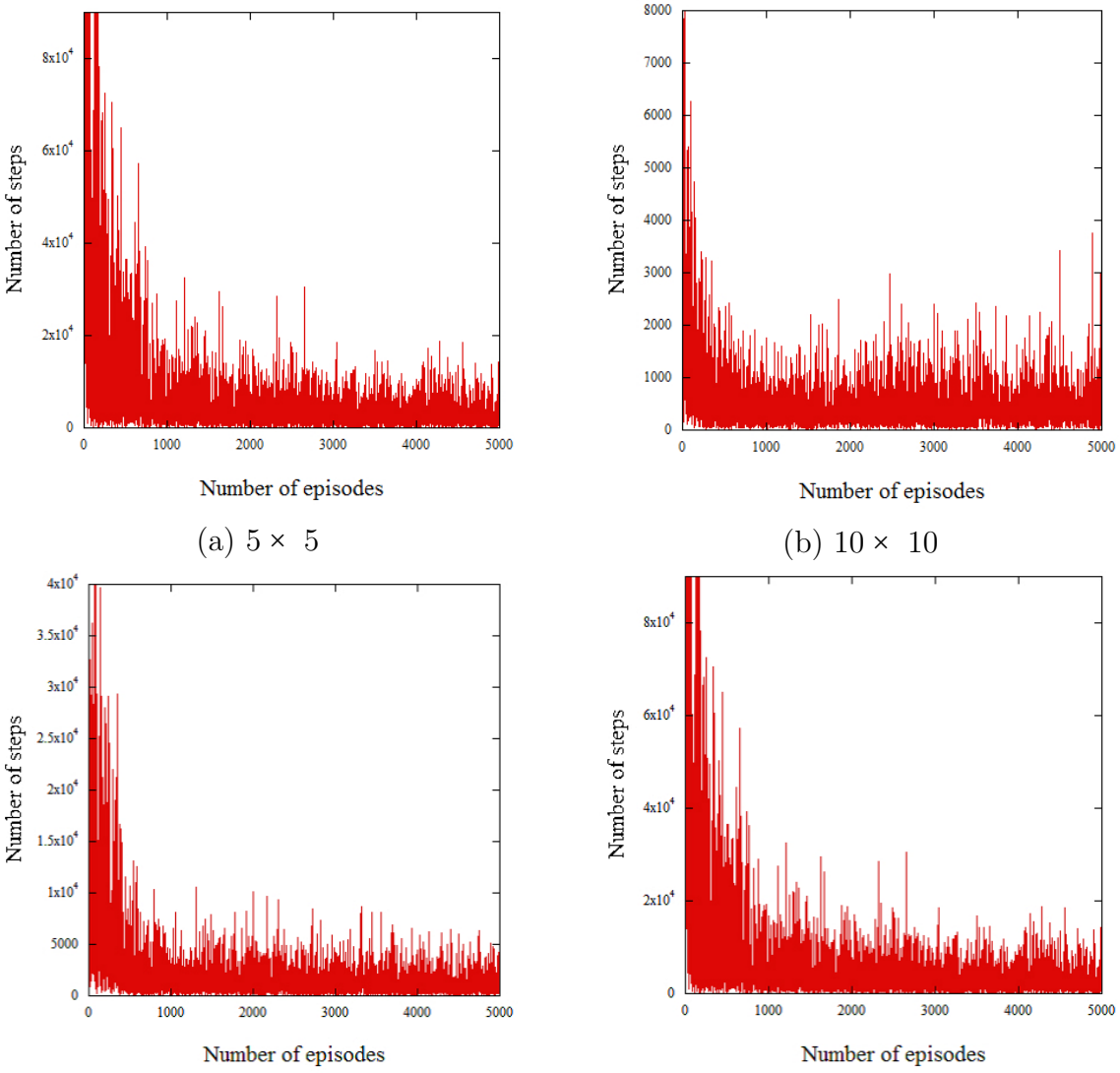
フラクタル次元の推移（図 5.14）では，全てのグリッドワールドサイズのにおいて，学習の初期のステップ数からの低下と，安定が現れている．これらの結果から，収束傾向が現れた学習曲線のフラクタル次元は全て 1.4 次元以下と言える．

5.6.2.4　非学習環境における結果

非学習環境において，各グリッドワールドサイズの学習曲線を 1 つずつ図 5.15 (a) から (d) に示す．また，それらの学習曲線の CEFDi によりフラクタル次元解析した結果を図 5.16 に示す．

非学習環境における学習曲線は，これまでの学習曲線と異なりハンターが学習を行わないため，収束傾向は現れない．これは，ハンターがランダムに行動し偶然獲物を捕獲したステップ数が出力され続けるからである．図 5.15(a)∼(d) から，グリッドワールドサイズによるステップ数の違いはあるが，学習曲線の傾向は同じである．

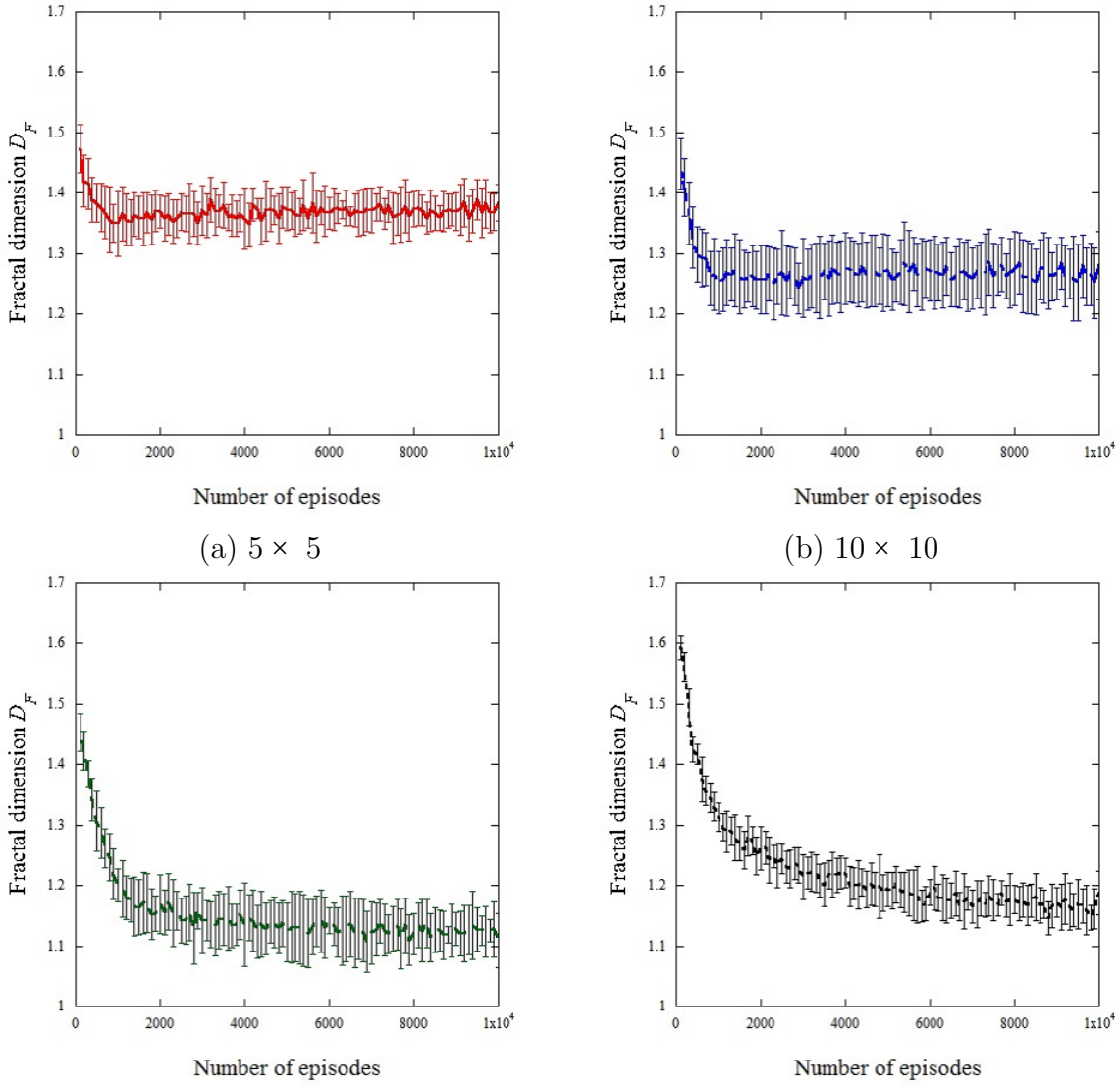
これらの学習曲線のフラクタル次元は，学習曲線と同様に横這い傾向が現れている．さらに，これまでのフラクタル次元の推移と異なり 1.4 次元から 1.5 次元の間で推移を維持している．また，標準偏差もフラクタル次元の推移と同様に横ばいとなっている．



(a) 5 × 5　　(b) 10 × 10

(c) 15 × 15　　(d) 20 × 20

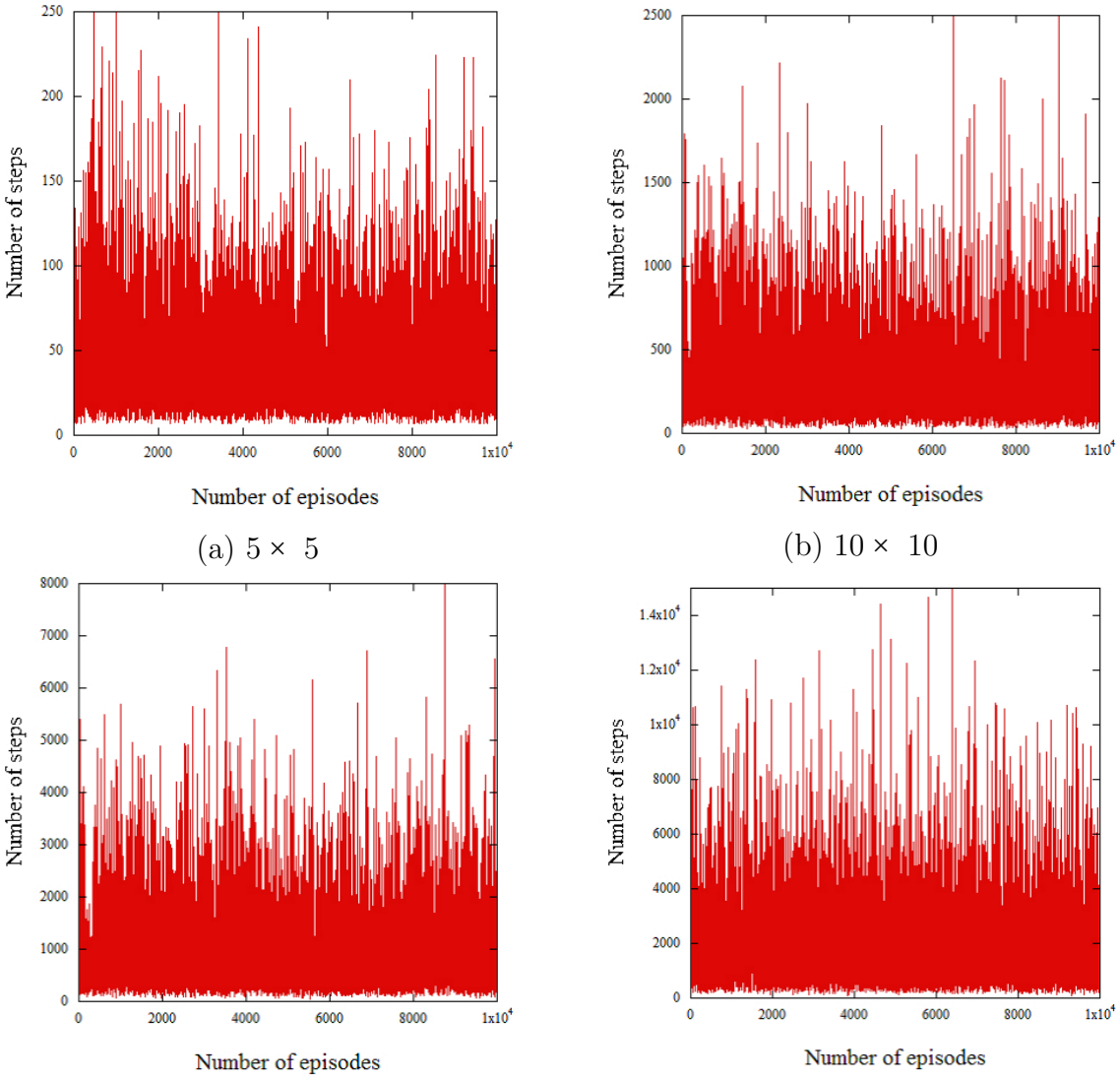
図 5.13 マルチハンター環境における各グリッドワールドの学習曲線（Trial 1）



(a) 5 × 5　　(b) 10 × 10

(c) 15 × 15　　(d) 20 × 20

図 5.14 マルチハンター環境におけるフラクタル次元解析結果



(a) 5 × 5　　(b) 10 × 10

(c) 15 × 15　　(d) 20 × 20

図 5.15 静的環境における各グリッドワールドの学習曲線（Trial 1）

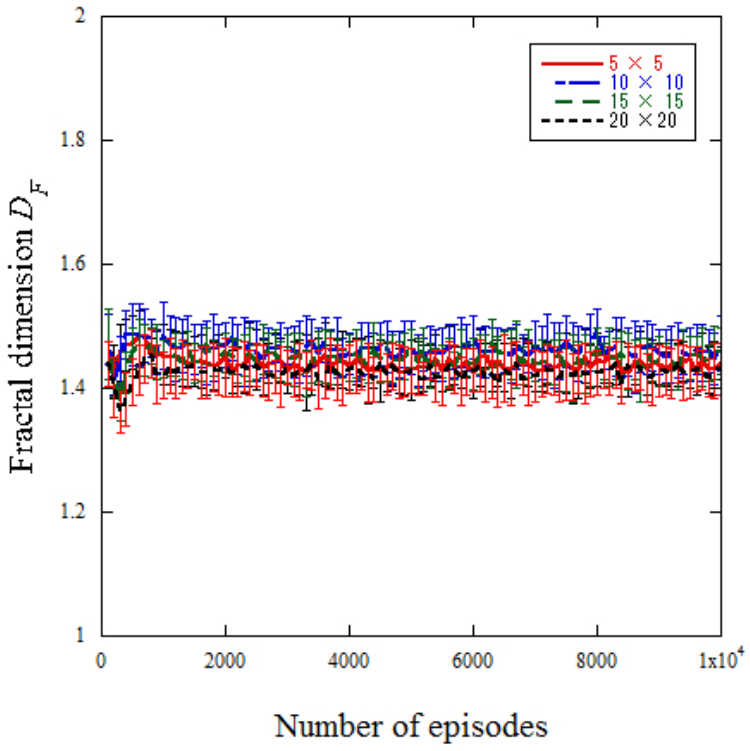


図 5.16 非収束環境におけるフラクタル次元解析結果

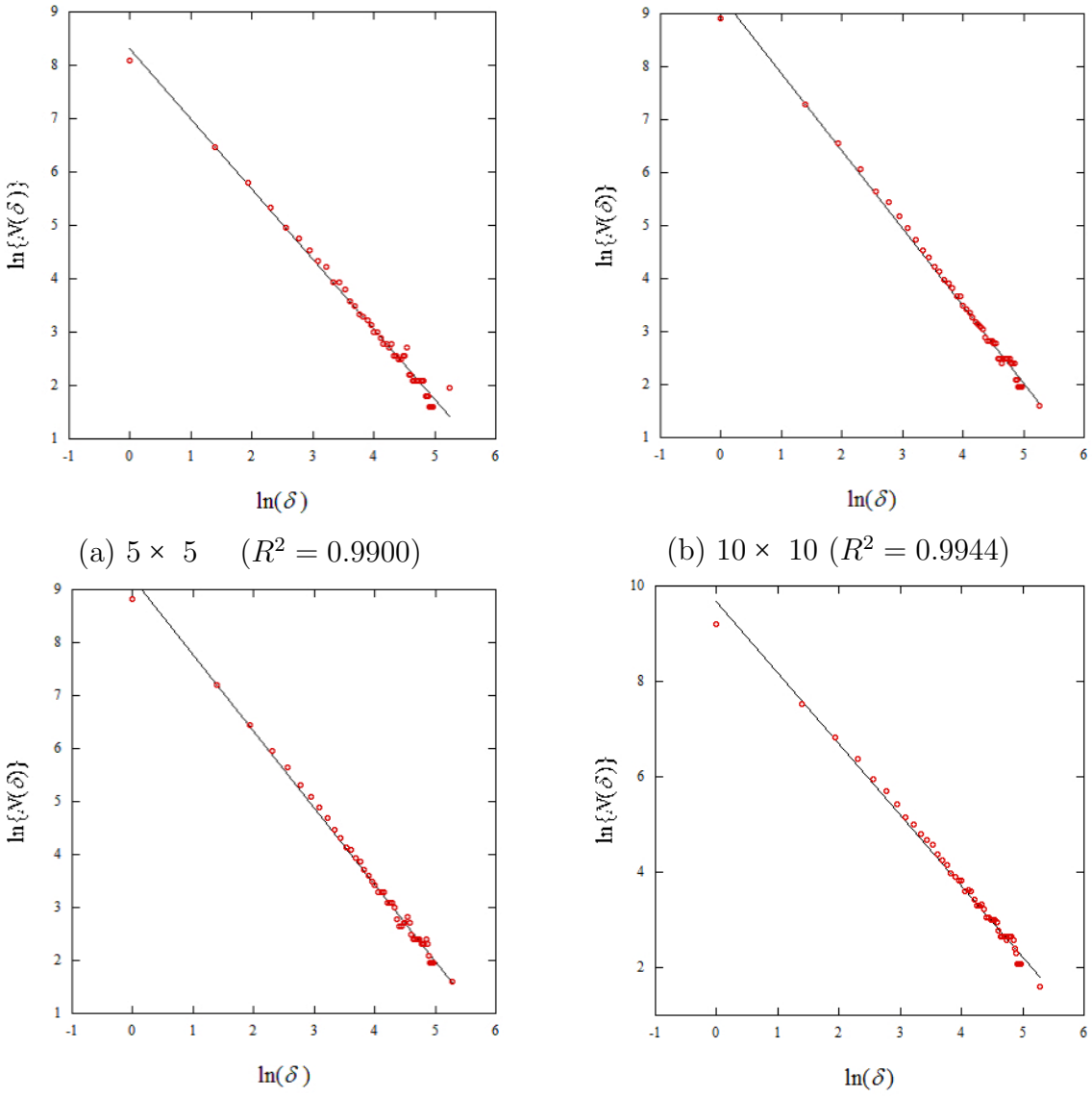
5.6.3　考察

5.6.3.1　学習曲線のフラクタル性

本実験では，学習曲線にフラクタル性が存在するという仮定の基，実験を行った．ここでは，実験から得られたデータよりフラクタル性の存在を考察する．

各グリッドワールドサイズにおける，図 5.2 に示したようなフラクタル次元算出時に用いる log-log グラフを図 5.17 から図 5.20 に示す．これらの図は各実験条件 10 trial 中の 1 trial 目 1 episode から 100 episode をサンプルとして示している．これは学習曲線の試行錯誤過程におけるベキ分布との相関を意味する．また，図 5.17 から図 5.20 の各グラフにおける相関係数 R^2 を表 5.4 に示す．

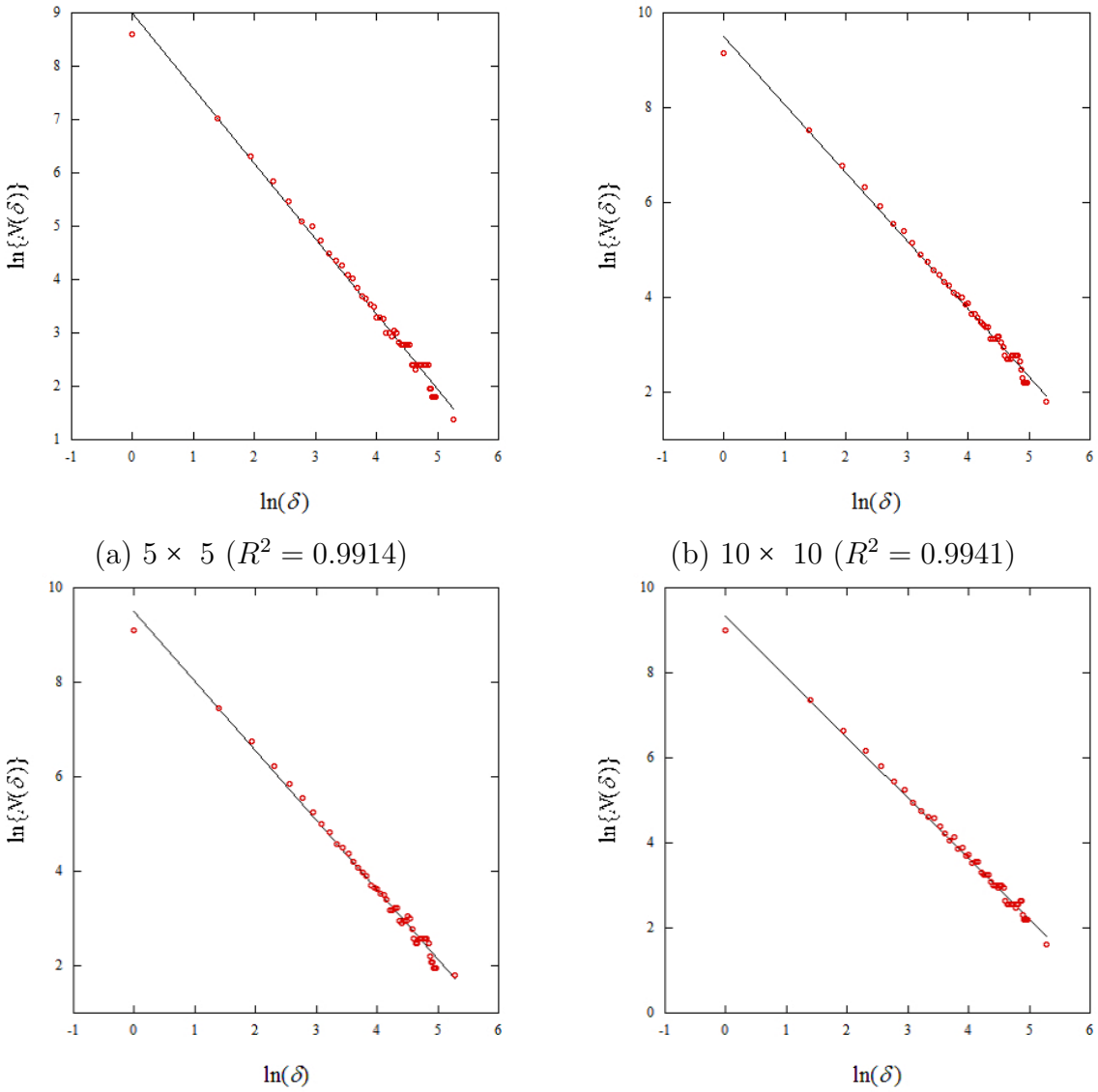
これらの解析結果から，どの学習曲線も試行錯誤過程においてはベキ分布との相関係数が少なくと約 0.99 以上あることがわかる．したがって，学習曲線は式 (5.4) に示した自己相似性を十分に持っていると考えられる．



(a) 5 × 5 ($R^2 = 0.9900$)　(b) 10 × 10 ($R^2 = 0.9944$)

(c) 15 × 15 ($R^2 = 0.9947$)　(d) 20 × 20 ($R^2 = 0.9930$)

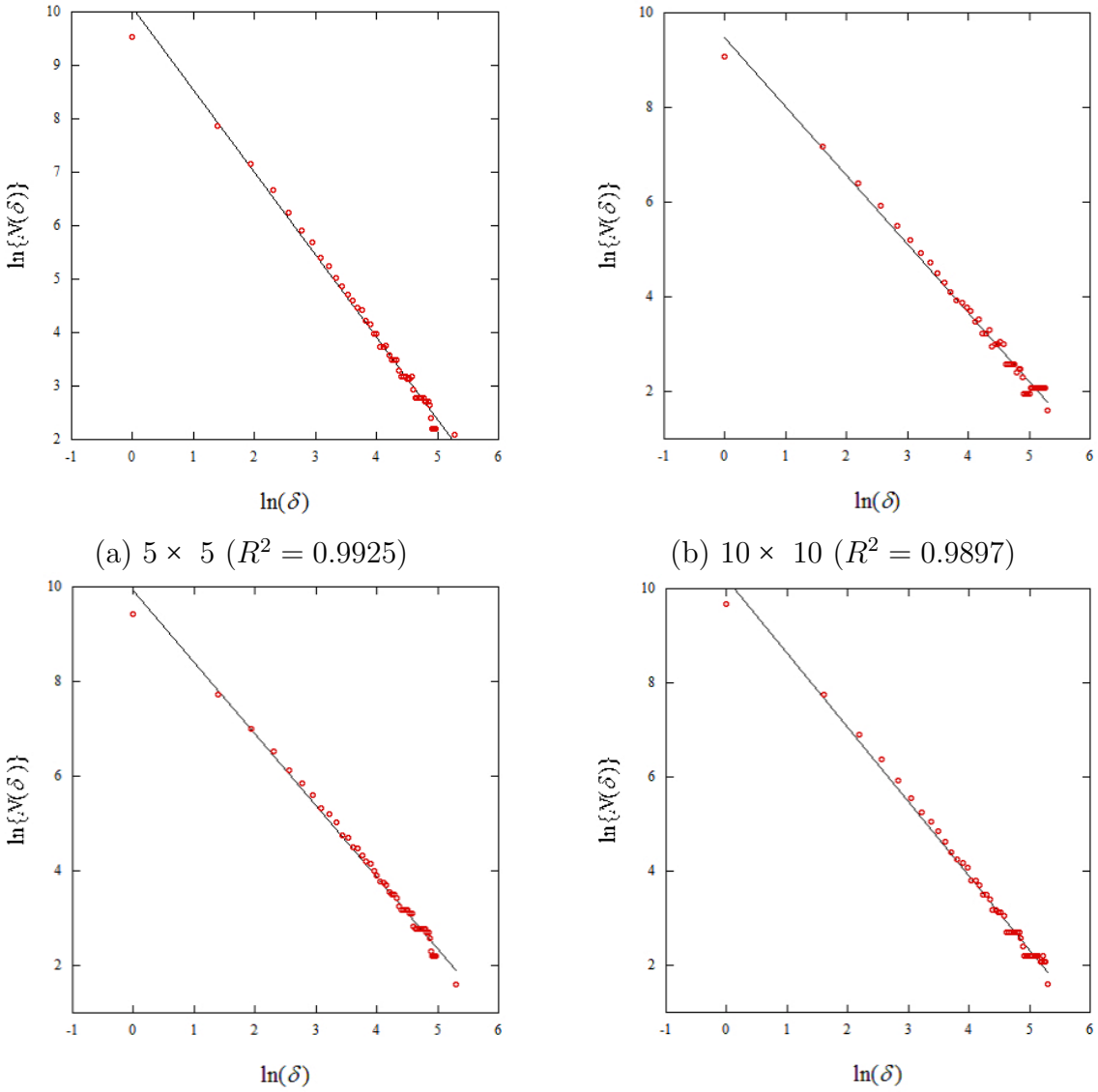
図 5.17 静的環境におけるベキ分布との相関（1∼100episode）



(a) 5 × 5 ($R^2 = 0.9914$)　(b) 10 × 10 ($R^2 = 0.9941$)

(c) 15 × 15 ($R^2 = 0.9926$)　(d) 20 × 20 ($R^2 = 0.9944$)

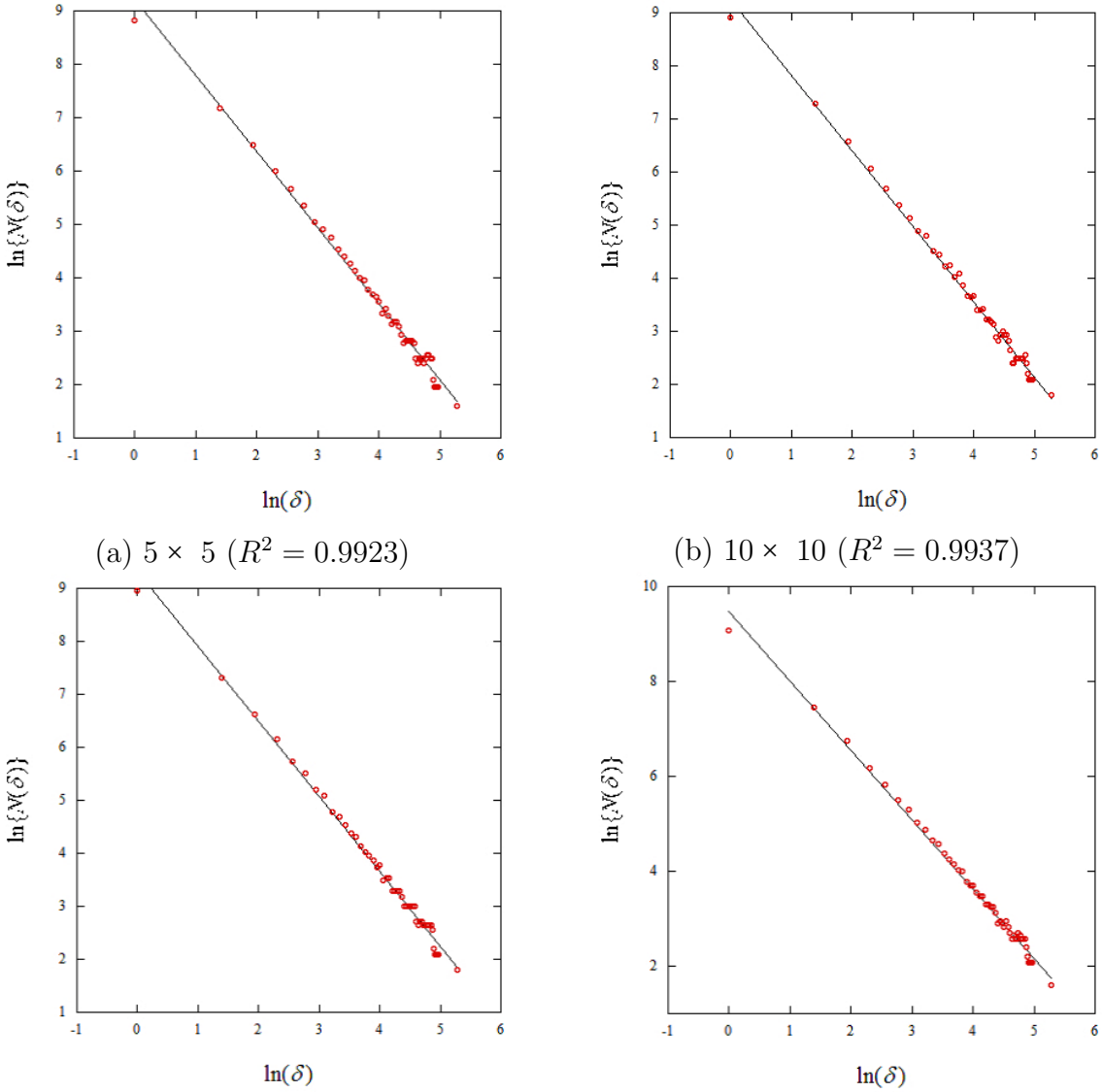
図 5.18 動的環境におけるベキ分布との相関（1∼100episode）



(a) 5 × 5 ($R^2 = 0.9925$)　(b) 10 × 10 ($R^2 = 0.9897$)

(c) 15 × 15 ($R^2 = 0.9923$)　(d) 20 × 20 ($R^2 = 0.9926$)

図 5.19 マルチハンター環境におけるベキ分布との相関（1∼100episode）



(a) 5 × 5 ($R^2 = 0.9923$)　(b) 10 × 10 ($R^2 = 0.9937$)

(c) 15 × 15 ($R^2 = 0.9932$)　(d) 20 × 20 ($R^2 = 0.9944$)

図 5.20 非収束環境におけるベキ分布との相関（1∼100episode）

表 5.4 CEFDi の計算機実験における相関係数 R^2（1∼100episode）

Conditions		R^2
Static	5×5	0.9900
	10×10	0.9944
	15×15	0.9947
	20×20	0.9930
Dyanamic	5×5	0.9914
	10×10	0.9941
	15×15	0.9926
	20×20	0.9944
Multi-hunter	5×5	0.9925
	10×10	0.9897
	15×15	0.9923
	20×20	0.9926
Non convergence	5×5	0.9923
	10×10	0.9937
	15×15	0.9932
	20×20	0.9944

5.6.3.2　収束推定のための閾値設定

静的環境において，全ての学習曲線が一意のステップ数に収束し，フラクタル次元もグリッドワールドサイズの違いは現れなかった．学習曲線が収束するまでのハンターの試行錯誤過程では，フラクタル次元推移の傾きや標準偏差に違いはあるが，収束を推定するには 1 次元，あるいは誤差を許容するとしても 1.05 次元など厳しい閾値による収束推定が可能であると考えられる．

動的環境において，収束傾向が現れているグリッドワールドサイズ 5 × 5 と 10 × 10 では，それぞれ 1.3 次元，1.4 次元を下回っている傾向があるため，収束を推定する閾値として 1.4 次元が利用可能であると考えられる．また，収束傾向が現れているが 10000episode を経過してもパフォーマンスの改善が見られるグリッドワールドサイズ 15 × 15 と 20 × 20 においては，閾値を 1.4 次元と設定し収束を推定すると学習途中で収束と推定し，誤判定の可能性も考えられる．10000episode 以上で収束することを期待すれば，1.3 次元程度に閾値を設定することも考えられる．

マルチハンター環境では，全ての学習曲線で収束傾向が発現し，フラクタル次元においても推移の安定が見られる．全ての学習曲線を 1 つの閾値で収束推定を行うとすれば，1.4 次元に設定できる．しかし，グリッドワールドサイズ 15 × 15 以上では，閾値を 1.4 次元と高めに設定すると，学習途中で収束を推定してしまう可能性があることに注意が必要である．

非学習環境においては，実験結果からフラクタル次元が 1.4 次元と 1.5 次元の間で推移する．より複雑な環境では学習曲線のフラクタル次元が高くなる可能性も考えられる．すなわち，非収束な学習曲線におけるフラクタル次元の閾値は 1.4 次元と設定し，1.4 次元以上の学習曲線は収束しない，もしくは収束していない学習曲線であると考えられる．

5.6.3.3　フラクタル次元の推移における環境依存性

これまでは，フラクタル性と収束推定のための閾値を議論した．ここでは，環境の違いに対する収束推定への影響を考察する．

静的環境において，全ての学習曲線で異なるステップ数が得られたのに対し，収束時におけるフラクタル次元の大きな違いが現れない．これは，CEFDi が学習曲線のステップ数を評価しているのではなく，波形の形状を次元数として定量化し評価しているためである．異なるステップ数の推移を，フラクタル次元で再構成することで，1 次元から 2 次元の間で評価することを可能にしている．この静的環境における結果は，最も単純かつ直感的に理解しやすい結果である．

一方，動的環境ではグリッドワールドサイズで極端にステップ数の変化している学習曲線を，静的環境同様に 1 次元から 2 次元の間で表現することが可能である．さらに，高いフラクタル次元から収束と共に低いフラクタル次元へ推移していることから，異なる環境でもフラクタル次元解析と，任意の閾値により収束の推定が可能であることが示唆される．しかし，各グリッドワールドサイズは異なるフラクタル次元へ収束し，閾値のみでは収束推定が困難であると考えられる．

マルチハンター環境も動的環境と同様に，極端にステップ数の異なる 4 つの学習曲線を，1 次元から 2 次元の間で表現できている．しかし，全ての学習曲線は収束傾向が現れているが，異なるフラクタル次元へ収束しているため，一意の閾値では収束が推定できない可能性がある．また，動的環境とマルチハンター環境の傾向として，グリッドフィールドサイズが大きいほど学習曲線の収束時のフラクタル次元は低くなる傾向があると考えられる．これは，狭いグリッドワールドでは試行錯誤過程のステップ数と収束時のステップ数に差が出来にくく，学習曲線の複雑性という面では比較的違いが表れにくいこ

とが原因であると考えられる．

非学習環境においては，グリッドワールドのサイズに関係なくフラクタル次元の推移が発現してる．これらの結果から，CEFDi はグリッドワールドサイズの大きさや，エージェントの数などに起因する環境の複雑性に依存しにくい収束推定法であることが示唆される．しかし，収束を推定する閾値は環境に応じて微調整が必要であるとも考えられ，また，他の評価指標（例えば尤度など）が必要であると考えられる．

5.7 CEFDn の計算機実験

本実験では，計算機実験により CEFDn の有用性を評価する．前節の実験と同様に，本実験でも強化学習エージェントによる追跡問題を採用し，静的環境や動的環境を用いて数値計算を用いた CEFD の有用性を評価する．

CEFDi は，CEFDn と異なりエージェントが強化学習を行いながら同時に収束推定する．そのために，本実験では，CEFDi と CEFDn のフラクタル次元の算出精度や，収束推定時におけるフラクタル次元の算出頻度を検討することが目的である．

5.7.1 実験条件

本実験では，以下の 2 種類の実験を行い，CEFDn のフラクタル次元計算誤差と推定精度を評価する．

(1) CEFDi と CEFDn のフラクタル次元計算比較
(2) 収束推定間隔の検証
(3) 学習中の収束推定の検証

(1) では，前節の実験から得られた学習曲線を用いる．学習曲線は，図 5.21 に示す静的環境と動的環境，マルチハンター環境，非収束環境の 4 種類を用いる．CEFDi と CEFDn により，それらのフラクタル次元を計算し，計算結果の違いを明らかにする．CEFDi により計算したフラクタル次元を基準とし，CEFDn のフラクタル次元にどれだけ誤差が発生するのか確認する．また，本実験では CEFDi のフラクタル次元解析ソフトウェアに Fractal dimansion estimator を利用した．

(2) の実験では，CEFDn による収束推定が何エピソードに 1 回実行すればよいのか，実験的に求める．収束推定間隔は，毎エピソードごとに行うことが理想であるが計算負荷を高める可能性も懸念され，計算する頻度は少ない方が良いことは直観的にも理解できる．本実験では，収束推定間隔 p を 1，10，50，100 に設定しフラクタル次元の推移を考察する．本実験では，学習可能なハンター 1 台と行動可能な獲物 1 台の動的環境を用い，追跡問題を用いて実験を行う．グリッドワールドの大きさは 7 × 7 とする．学習パラメータは表 5.5 に示す．

(3) の実験は，(2) と同様に追跡問題を用いて実験を行う．本実験は，追跡問題でハンターが捕獲行動を学習中に収束の推定を行い，十分に学習した状態のパフォーマンスと比較することで収束推定の精度を確認することが目的である．前節の実験結果より，単

表 5.5 CEFDn の計算機実験におけるパラメータ設定

Parameter	Value
Learning rate	0.1
Discount rate	0.9
Reward	1
Boltzmann parameter	0.01
Default Q-value	0
Number of episode	1000
Number of trial	10

純に閾値を用いただけでは正確な収束推定が行えない可能性がある．そこで，本実験ではフラクタル次元連続更新回数 N というパラメータを導入する．

収束推定の基準として，学習曲線のフラクタル次元が閾値 TH を下回り，かつフラクタル次元の下限値を連続して N 更新しなかったときを収束とする手法である．これは，フラクタル次元が N 回下限値を更新しない場合は，これ以上のパフォーマンス改善は見込めないという着想の下，導入したパラメータである．

本実験では，フラクタル次元連続更新回数 N を 100，200，300，400，500 に設定して実験を行い，10 回試行して結果を考察する．

5.7.2 実験結果

5.7.2.1 CEFDi と CEFDn のフラクタル次元算出誤差

CEFDi と CEFDn により 4 種類の学習曲線をフラクタル次元解析した結果を図 5.22 ～図 5.25 に示す．これらの図形は，実線が CEFDi であり破線が CEFDn である．これらの結果から，CEFDn は画像処理による CEFDi と同様の傾向の次元数が算出されていることがわかる．

5.7.2.2 収束推定間隔の結果

本実験では，$p = 1, 10, 50, 100$ の 4 種の収束推定間隔により比較を行った．また，本実験で獲得した学習曲線は図 5.26 に示す．収束推定間隔 p の各フラクタル次元の推移を図 5.27(a)～(d) に示す．

この結果から，図 5.27(a) のように 1episode 間隔でフラクタル次元解析を行うと，学習曲線の特徴が最も細かく抽出することが可能である．しかし，図 5.27(b)∼(d) のように間隔をあけて解析を行うと，解析する間隔の間の中間点が抜け落ちる波形の推移となる．学習の収束はフラクタル次元の値を直接参照し判断するため，短い間隔でフラクタル次元が算出される方が望ましい．したがって，短い間隔で解析すれば計算量の削減を目的として提案した CEFDn でも，計算負荷が高くなる可能性も考えられるが，CEFDn は 1episode 間隔ごとにフラクタル次元解析を行うことが望ましいと言える．

5.7.2.3　学習中の収束推定の結果

フラクタル次元連続更新回数 N に対する算術平均と標準偏差を図 5.28 に示す．この結果から，N の値を大きくすると，平均や標準偏差が小さくなり収束推定誤差が小さくなることがわかる．しかし，$N = 500$ でも最終的な平均ステップ数と比較して約 10%程度のステップ数の誤差が生じている．これは，実際に収束する直前で収束推定してしまう可能性を示唆する．さら N を大きくすることで，この誤差は小さくなると予想されるが，対照的に 500 エピソード以上ものフラクタル次元の値の更新を監視し続ける必要があり，収束推定が確定するまでに時間を要する．よって学習時間と誤差の軽減はトレードオフの関係であり，本実験の結果から両者を考慮し $N = 300$ が良好な値である言える．

5.7.3　考察

これらの実験結果から，CEFDi と CEFDn は実験的に同傾向，同等のフラクタル次元の推移を算出可能であり，さらに CEFDn の実行間隔を毎エピソードごとにすることで収束を推定するアルゴリズムとして有用であることが示せた．

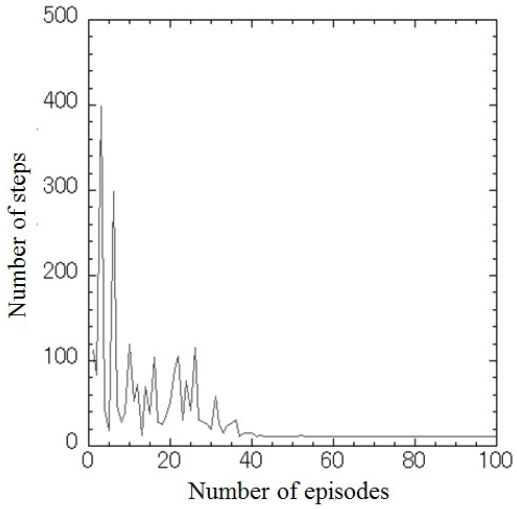
また，フラクタル次元連続更新回数 N は今回の実験結果より 300 とし，さらに前節の実験結果より閾値を導入することで学習中の収束推定が可能であると考えられる．例として，前項の実験で得られた学習曲線を以下の条件で収束推定する．

- $p = 1$
- $N = 300$
- $TH = 1.5$

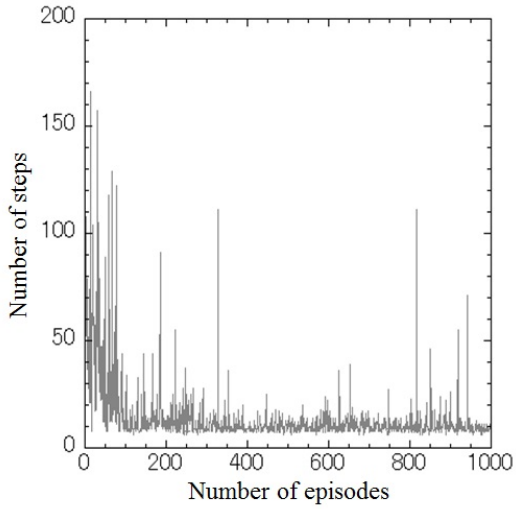
この条件にて収束推定を行うと，1585 episode 目に収束と推定された．学習曲線に対する 1585 episode を図示すると図 5.29 となる．図 5.29 は，動的環境であるため収束傾向が

表れている曲線でも多少のスパイクのような波形が出力されている．しかし，学習初期の高いステップ数からエピソードを継続し 1585 episode で収束を推定した以降もパフォーマンスは同等であると直観的に見て取れる．

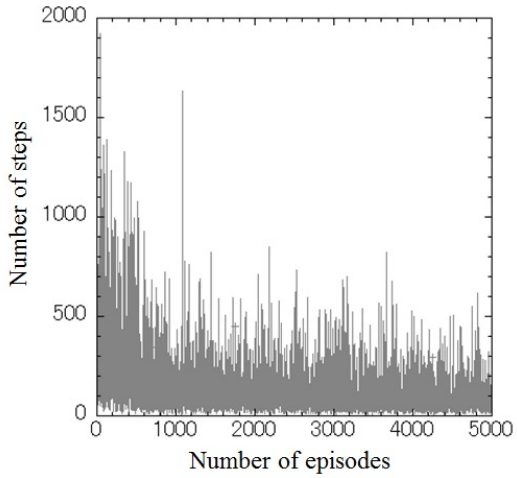
ここで，具体的なステップ数を評価する．任意のエピソード n におけるステップ数を S_n とする．収束が推定されたエピソードを i とし，その直前 10 episode のステップ数平均 $\frac{1}{10}\sum_{n=i-9}^{i} S_n$ と，全エピソード終了時における 10 episode のステップ数平均 $\frac{1}{10}\sum_{n=9991}^{10000} S_n$ を比較する．収束した，1585 episode のステップ数平均は 8.8 step であり，十分に学習した 10000 episode のステップ数平均は 9.0 step であった．これらの差は 0.2 step であり，誤差率では 2.2%であった．この結果から，動的環境においても CEFDn により収束の推定が可能であることが示唆された．



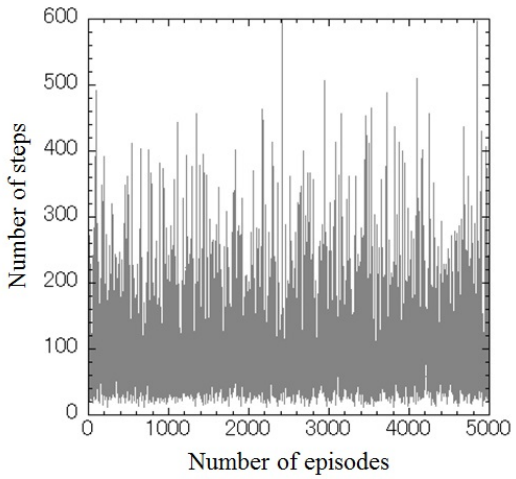
(a) Static environment



(b) Dynamic environment



(c) Multi-hunter emvironment



(d) Non convergence environment

図 5.21 CEFD の比較に用いる学習曲線

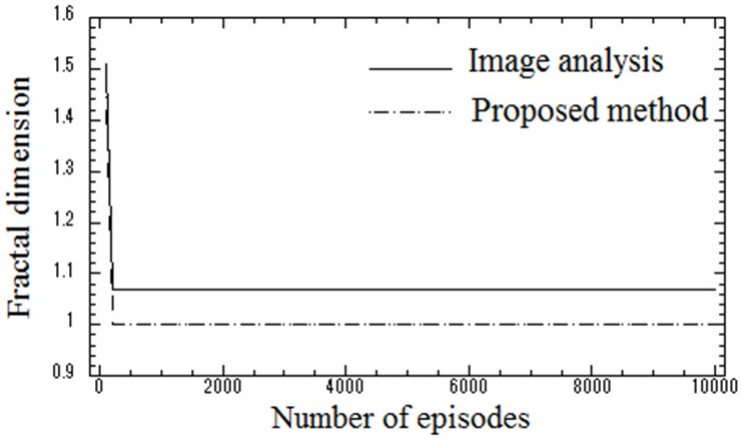


図 5.22 静的環境におけるフラクタル次元算出誤差

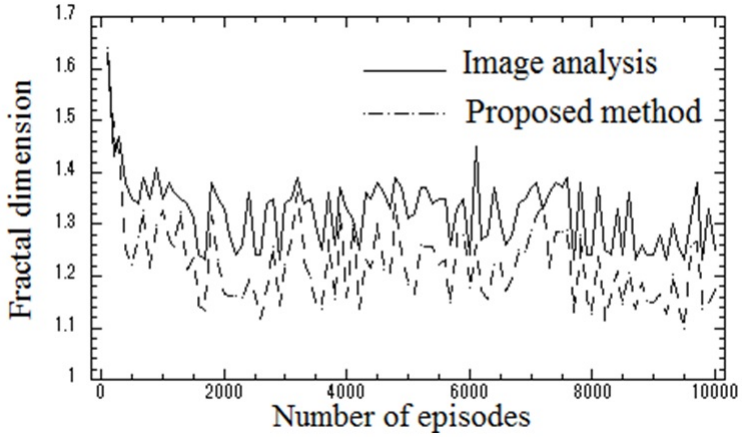


図 5.23 動的環境におけるフラクタル次元算出誤差

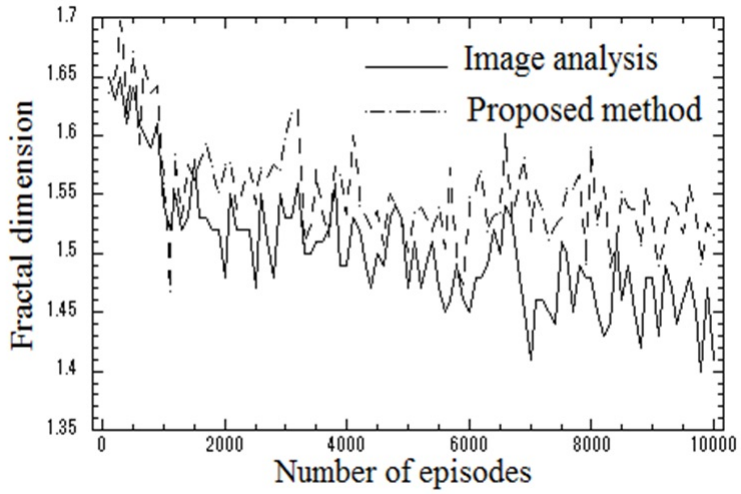


図 5.24 マルチハンター環境におけるフラクタル次元算出誤差

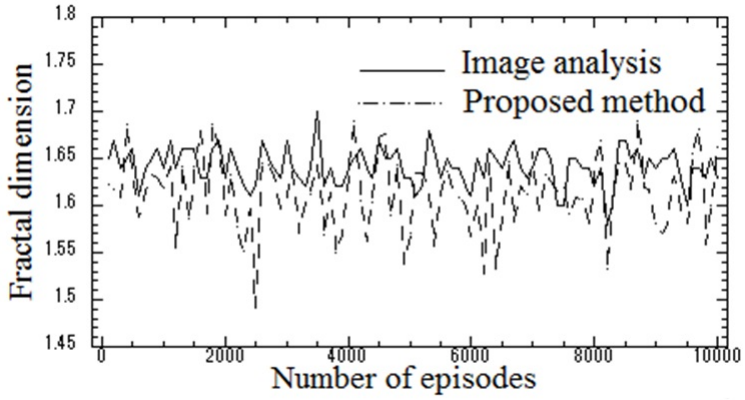


図 5.25 非学習環境におけるフラクタル次元算出誤差

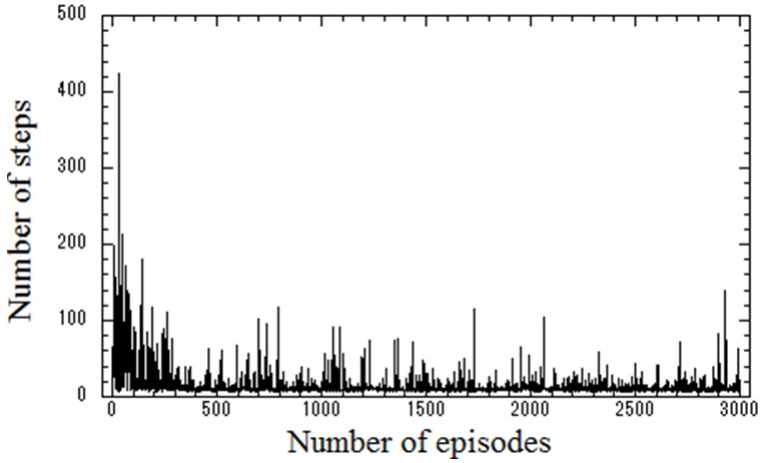
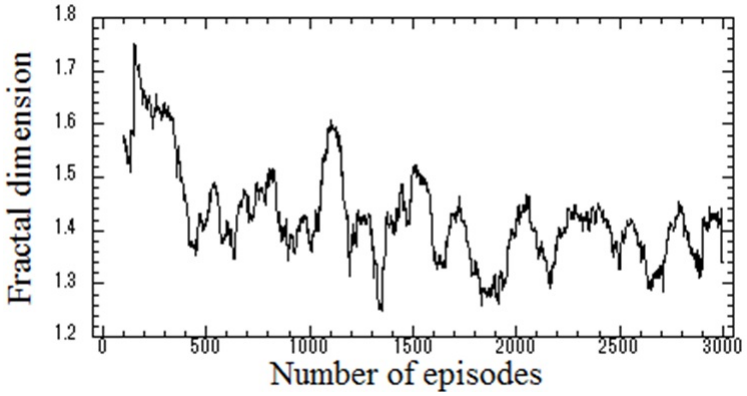
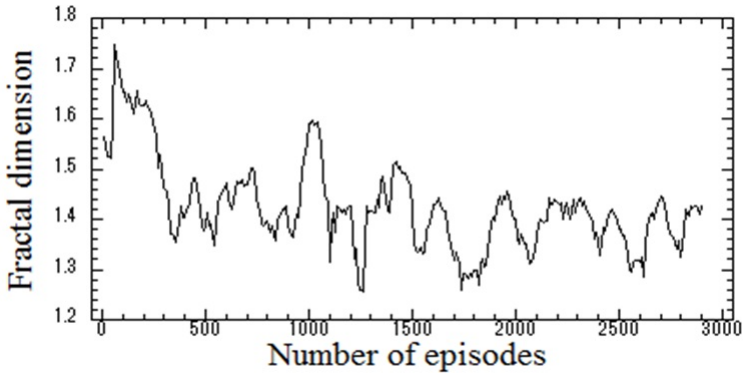


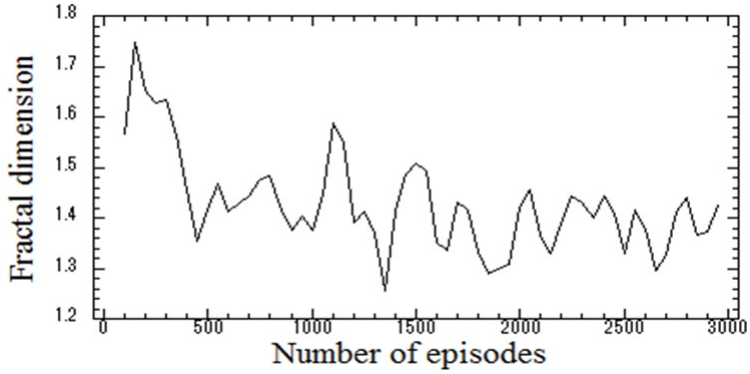
図 5.26 CEFDn により解析した学習曲線



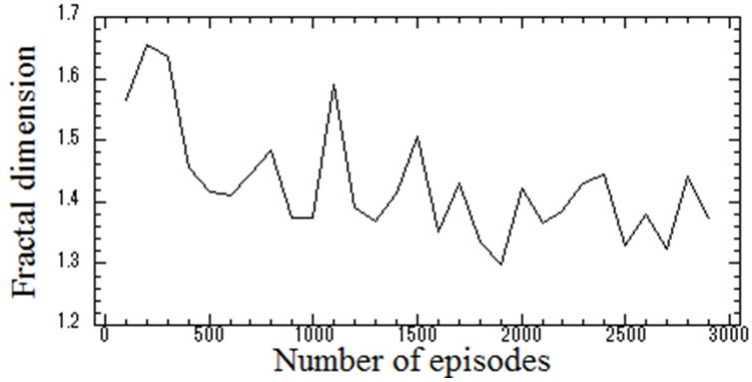
(a) $p = 1$



(b) $p = 10$



(c) $p = 50$



(d) $p = 100$

図 5.27 解析実行間隔 p によるフラクタル次元の推移

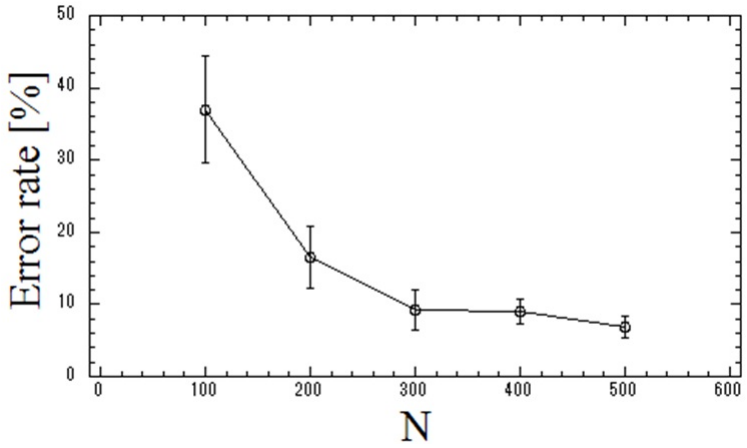


図 5.28 フラクタル次元連続更新回数 N の収束推定誤差率

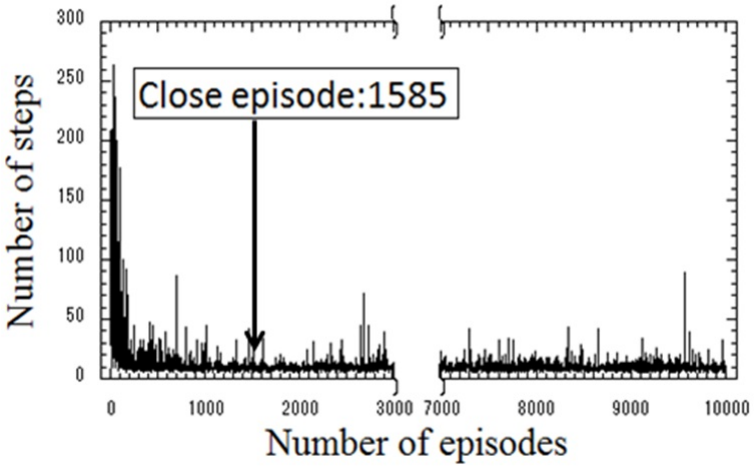


図 5.29 学習曲線における収束推定エピソード

5.8 結言

本章では，エージェントが強化学習による知識の獲得を自律的に判定する手法について議論した．

上述の議論に対して，フラクタル次元解析により学習曲線を解析し，収束を推定する CEFD を提案した．初めに，学習曲線がフラクタル性を有しているとの仮定の下，画像解析によるフラクタル次元解析を用いた CEFDi を提案し，計算機実験から閾値による収束推定が可能であることを示唆した．さらに，学習曲線が完全ではないがフラクタル性を有していることを示した．また，次に画像処理を用いない近似的なフラクタル次元解析を用いた，数値処理によるフラクタル次元解析を実現する CEFDn を提案し，計算機実験から有用性を示した．これらの結果より，以下の成果が得られた．

(1) 学習曲線の試行錯誤過程にフラクタル性が存在することが示唆された．

(2) 静的環境や動的環境に存在するエージェントが，ステップ数や収束タイミングが異なる学習曲線を CEFDi により収束の推定が可能であることを計算機実験により確認した．

(3) エージェントへの実装を考慮した CEFDi による収束推定を用いて，閾値 TH や収束推定間隔 p，フラクタル次元連続更新間隔 N を適切に設定することで，エージェントが自律的に収束を推定可能であることを計算機実験により確認した．

上記の成果より，CEFDn をエージェントに実装することで，これまで実現していなかったエージェント自身による強化学習の収束推定が可能であると考えられる．これにより，知識の保存や共有のために，エージェントが十分なパフォーマンスを強化学習により獲得したことを自律的に判断し，学習を終了することが可能となる．

本論文で提案している知識共創フレームワークへの応用としては，エージェントが獲得した知識を自動的にクラウドへアップロードするなどの利用が可能となる．

第6章　外部計算機資源を用いた獲得知識の統合

6.1　緒言

本章では，複数台の強化学習エージェントが獲得した知識を，クラウド上で統合，共有する手法に関して議論を行う．表 1.1 に示した通り，本章は 1.4 節に提示した以下の議論点に対してアプローチを行う議論である．

- 獲得知識の処理に関する議論

これまで強化学習などの学習アルゴリズムでは，学習し知識を獲得することが目的であり，その獲得した知識を別途保管し管理，運用する手法に関しては議論が深くなされてこなかった．近年は，転移学習の議論の活発化や，計算機性能の向上によるクラウドの利用により，学習エージェントが獲得した知識を，外部に保管し，適宜利用するという方法が提案されている．

例えば，近年のクラウドロボティクスに代表される RoboEarth では，ヘテロジーニアス（類似）な自律移動ロボットがクラウドを介して知識を再利用し，迷路探査を行う実験が行われている [36]．しかし，多くのタスク，多くの強化学習エージェントが存在するような環境では，強化学習エージェントから生成される知識の量は増加すると考えられる．したがって，実世界応用を考慮すると，類似した知識は統合し，再利用できるように管理し，知識を再利用するエージェントが知識を選択してダウンロードできるようなメカニズムが必要である．したがって，以下の課題を設定する．

課題：「エージェントが獲得した知識を処理するメカニズムが必要」

本章ではクラウドを用いた強化学習エージェントの獲得知識処理手法を提案する．これにより，インターネットに接続可能な強化学習エージェントであれば，クラウドにアクセスし，タスクやエージェントの種類に応じた知識をダウンロード，すなわち再利用可能となる．

6.2 強化学習のための知識処理

本節では，強化学習エージェントが獲得した知識の統合や共有，選択手法など知識処理に関する関連研究を俯瞰する．次項より，各機能に分けて関連研究を記す．

6.2.1 知識の統合手法

サッカーロボットにおける衝突回避行動の知識と，シュート行動の知識を結合する研究が内部らにより行われている [98]．各知識は個別に学習され，後にそれらを統合しサッカーロボットに知識を再利用させ，両知識を組み合わせた行動が発現することを確認している．サッカーロボットはボールをゴールにシュートする行動と，ゴールキーパーとの衝突回避を個別に学習し，それらの知識をシンプルサムという単純に知識（Q テーブル）の行動価値を和算し，その知識を用いてサッカーロボットがキーパーを避けながら，ボールをゴールにシュートする行動の発現を確認している．シンプルサムは次式により定義されている．

$$ {}^{c}Q_{sw}({}^{c}s, a) = {}^{g}Q(({}^{g}s, *), a) + {}^{r}Q((*, {}^{r}s), a) \tag{6.1} $$

ここで，2 つの知識 ${}^{g}Q$，${}^{r}Q$ を加えることで，新しい知識 ${}^{c}Q_{sw}$ を構成する．$({}^{g}s, *)$，$(*, {}^{r}s)$ は統合後の状態を前の状態 ${}^{g}s$，${}^{r}s$ で表現するのに用いており，$*$ は任意の状態を表している．しかし，個別に学習する知識は明らかに目的の異なるタスクであり，同目的や類似目的の知識の統合まで考慮されていない．また，統合するそれぞれの知識において，学習進度が異なる場合，すなわち行動価値の差がある場合にエージェントは行動価値の高い行動を選択しやすいため，結合する知識のどちらかは使われない可能性が考えられる．

知識の統合と近いアプローチとして，Kretchmar は，マルチエージェントで同時並列的に Q 学習を実行し，そこで得られる報酬を統合・分配する Parallel rienforcement learning (PRL) を提案している [99]．PRL は複数台のエージェントの行動や得られる報酬を共有できることを前提として，次式に示す知識の統合方法を提案している．

$$ Q_t(a) = \frac{\tilde{Q}_t(a) * \tilde{k}_a + \hat{Q}_t(a) * \hat{k}_a}{\tilde{k}_a + \hat{k}_a} \tag{6.2} $$

ここで，$Q_t(a)$ は行動 a を実行した際に得られる報酬であり，自エージェントの報酬は $\tilde{Q}_t(a)$ である．$\hat{Q}_t(a)$ は他の前エージェントの報酬合計である．さらに，$\tilde{k}_a$ はエピソード（論文中ではトライアル）における自エージェントの行動 a の選択回数であり，$\hat{k}_a$ は他の

全エージェントの行動 a を選択した回数である．文献 [99] では，PRL を n 本腕バンディット問題に用いており，式 (6.2) はマルチエージェントが同様のスロットを並列に操作する環境を前提としている．Kretchmar の手法は，報酬の共有により結果的に他のエージェントの経験を取り入れた知識を獲得可能である．しかし，既に保存された複数個存在する知識を統合する手法ではなく，知識統合の応用へは議論が必要である．

6.2.2　知識の共有手法

知識の共有手法に関しては，第 1.3.2 章でも概説したが，クラウドコンピューティングを活用した方法がある．RoboEarth [36] では，自律移動ロボットが強化学習により，獲得した知識をクラウドで共有し，ヘテロジーニアスな自律移動ロボットがその知識を再利用する実験を行っている．しかし，タスクが迷路探査問題と 1 種であり，様々なタスクを学習し，なおかつ多台数ヘテロジーニアスな自律移動ロボットが存在する環境における，多くの知識を取り扱う手法が議論されていない．

6.2.3　知識の選択手法

高野は，禁止行動を用いた知識の選択手法を提案している [55]．禁止行動とは，予め知識を獲得するときに，エージェントにとって不都合な行動を意味し，負の報酬（マイナス値の報酬）で表現される．この禁止行動を獲得した知識と共に記録しておき，知識を再利用する場合，次式により禁止行動の一致率を計算し信頼度 C とする．予め設定された閾値 θ を超える知識を選択することで知識選択を実現している．

$$\text{信頼度} : C = \frac{\text{禁止行動の一致率}}{\text{全状態数}} \tag{6.3}$$

6.3　クラウドを用いた知識処理

前節までに示した関連研究をふまえ，本節では強化学習エージェントが，獲得した知識を共有・再利用するために，クラウドを用いた知識の統合手法を提案する．

知識を共有し，統合した後に，エージェントはタスクに応じて適切な知識を選択し，再利用しなければならない．そのため，知識の共有手法や統合手法，選択手法を個別に議論し検討する必要がある．本節では，まず知識処理の基礎的検討として，知識の統合手法に関して議論を行う．

6.3.1　知識の保存

強化学習において，エージェントが獲得した知識は以下の形式で保存されることが想定される．

- 獲得した知識のそのままの形式（Look-up table）．
- 関数近似手法により軽量化，抽象化された知識の形式．

獲得した知識を関数近似する手法は，過去の研究や本研究でも有用性が有用性が示されているが，知識を近似することで，近似誤差や情報の欠損が発生し，本来の知識を再現できない可能性も考えられる．そこで，本章では基礎的検討として，システムで取り扱う知識を，関数近似されていない Look-up table の状態のまま処理することを前提とする．

6.3.2　知識の統合処理

内部らの研究 [98] では，知識の統合を実現しているが，検証の結果から統合する知識は干渉しない知識，すなわち完全に分割された異なるタスクにおける知識の統合を実現している．異なるタスクの知識を統合して，それらの両知識に記述された行動の両方が発現することは，我々に有益な効果と知見をもたらす．しかし，実問題として統合処理する知識は必ずしも干渉しない独立した知識であるとは限らない．例えば，本章の冒頭でも少し述べたのように，同様のタスクで獲得した知識を全て保存しておくのではなく，それらを統合して知識量の削減を要求する場面も容易に考えられる．

また，統合する知識は学習進度が異なる場合も考えられる．統合する知識で，行動価値の差がある場合，エージェントは行動価値の高い行動を選択しやすいため，統合する知識のどちらかは使われない可能性が考えられる．

そこで本研究では，式 (6.1) を改良し同目的異解法の関係にある知識を統合可能な次式を提案する．

$$Q^i(s, a_i) = \rho\{Q_1(s, a_i) + Q_2(s, a_i) + \ldots + Q_n(s, a_i)\} \tag{6.4}$$

ここで，$Q^i(s, a_i)$ はある環境状態 s における行動 a_i の結合された行動価値であり，$Q_n(s, a_i)$ はそれぞれ同目的異解法の関係にある知識である．n は知識の識別子であり $n \in N^+$ である．ρ は，統合された知識を割り引いて使用するためのパラメータであり，信頼率と呼ぶ．信頼率 ρ は，$0 < \rho \leq 1$ とする．また，式 (6.4) は各知識の行動の個数 i は同じであることを前提とする．各知識のスケーリングも行わずに統合する．

式 (6.4) は，ρ により統合した知識の行動価値を低下させ，強制的に他の行動の選択を促す，すなわち試行錯誤を発生させる．これにより，統合された知識をエージェントが活用し，同一タスクにおいては過去の知識を再利用する．さらに，類似タスクにおいては知識を活用しつつ試行錯誤させることを可能にする．

上述の知識の統合処理は，図 6.1 のようにクラウドコンピューティングを利用して実行することを想定している．知識を生成するエージェントは，クラウドと通信を行い獲得した知識をアップロードする．多台数のエージェントからアップロードされた知識は，類似したものは提案手法により統合し知識量削減や知識同士の補完，新たな知識の生成などを行う．

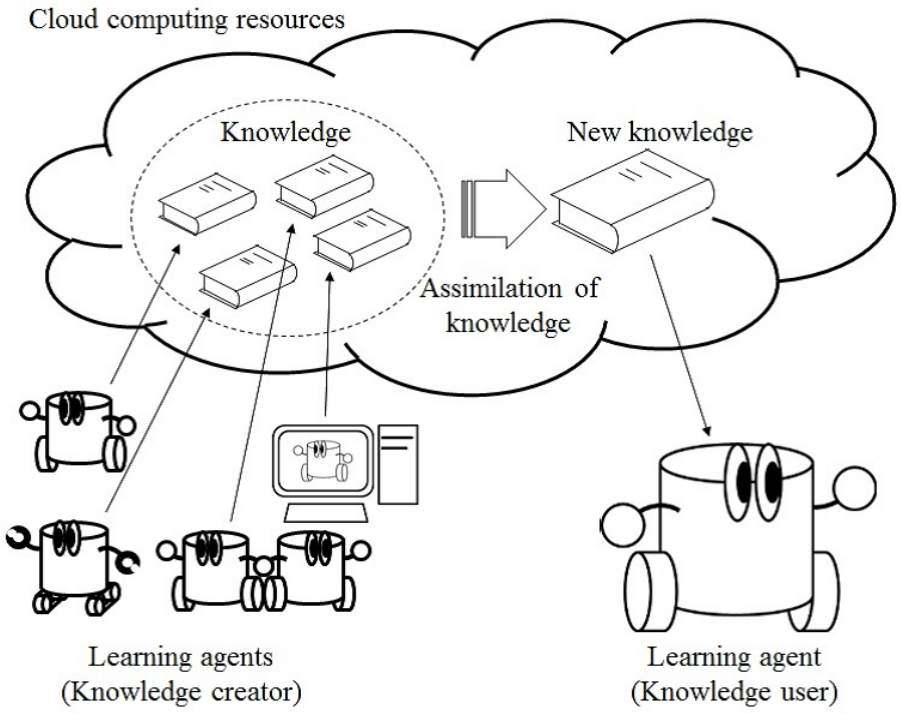


図 6.1 クラウドを活用した知識統合手法のイメージ

6.4　統合知識を用いた計算機実験

本実験では，知識の統合法を検証するために，基礎的検討として静的環境における追跡問題により評価する．これは，環境が MDP であるため事前に最適解が理解しやすいためである．本実験の目的は，2 つの異なる環境における同目的タスクで獲得した知識を，式 (6.4) により統合し，新たな環境で知識の再利用が可能か検証することである．

計算機実験を行うシミュレータ・プラットフォームとしては，付録 A のマルチエージェント強化学習シミュレータの設定を変更し，障害物有の最短経路問題により評価を行う．前章までの実験同様，数値シミュレーションを行う．

6.4.1　問題設定

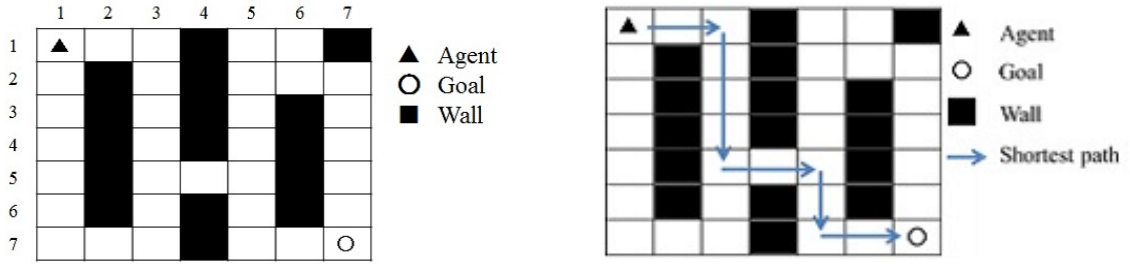
本実験では，シンプルなタスクにおける計算機実験にて，式 (6.4) の効果を検証する．タスクとしては，最短経路探査問題を採用する．最短経路探査問題は，強化学習エージェントが，ゴールに到達するまでの最短な経路を学習し，ゴールに到達するまでの時間や行動回数によりエージェントのパフォーマンスを評価する．本実験では，行動回数により評価し，すなわち学習曲線を評価する．以下より，本実験の設定や条件を述べる．

6.4.1.1　環境 · エージェント設定

本実験では，図 6.2(a) のように障害物を配置した，7×7 グリッドフィールドの最短経路探査問題を採用する．このグリッドフィールドを本実験の基本形とし，フィールド A と呼ぶ．フィールド A において，エージェントは壁を越えた移動が出来ない．さらに，エージェントは図 6.3 に示す移動特性を持ち，視界を持たず自己位置のみ認識可能である．フィールド A は 4 種類のゴールへの経路が存在する環境であるが，図 6.3 のエージェントを用いると，フィールド A の最短経路は図 6.2(b) のようになることは明らかである．すなわち，エージェントの初期座標を (1, 1) とすると，最短経路のステップ数は 12 ステップとなる．

6.4.1.2　実験条件

本実験では図 6.4 のように，フィールド A にさらに障害物を配置したグリッドワールドを用いる．フィールド B においては，2 つの障害物を座標 (2, 1) と (5, 3) に配置し，最短



(a) グリッドワールド　　(b) 最短経路

図 6.2 障害物を配置したグリッドワールドとその最短経路

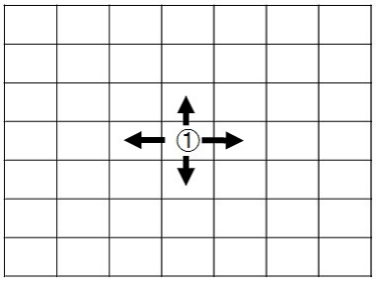


図 6.3 最短経路探査問題に用いる強化学習エージェント

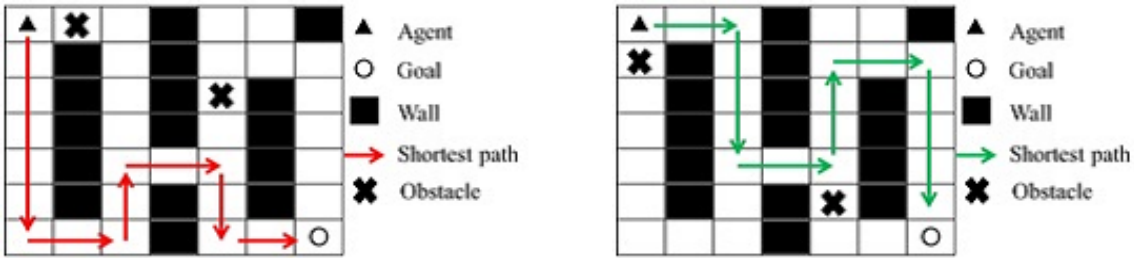
経路は図 6.4(a) の通りである．フィールド C においても，2 つの障害物を座標 (1, 2) (5, 6) に配置し，その最短経路は図 6.4(b) となる．

これらの障害物配置は，フィールド A が持つ複数の経路を封鎖し，強制的に 1 つの最短経路を学習させるためである．これにより，フィールド A とフィールド B，フィールド C はタスクの目的は同様であるが，解法が異なる知識を学習可能である．

ここで，学習パラメータや実験エピソード数などの値を表 6.1 にまとめる．また，本実験では知識の再利用時に転移学習を行うが，転移するエージェントはホモジーニアスであるため，ITM は Source task のエージェントと Target task のエージェントにおいて，同じ行動をマッピングしている．

6.4.2　実験結果

各信頼率における，エピソード数に対するステップ数を示した学習曲線を図 6.5 に示す．図 6.5 は，付録 B に示すスムージング処理を行いグラフの描画を行っている．$\rho = 0$ のときは，一から学習を行うため，学習曲線は高いステップ数から最短経路へ収束する．また $\rho = 0.8, 1.0$ において，知識 B の行動価値は知識 C と比較して高いために優先的に知識



(a) フィールド B の最短経路　　(b) フィールド C の最短経路

図 6.4 フィールド B とフィールド C の最短経路

表 6.1 統合知識を用いた計算機実験のパラメータ設定

Parameter	Value
Learning rate α	0.1
Discount rate γ	0.9
Reward r	1
Boltzmann parameter T	0.005
Default Q-value	0
Reliable rate ρ	0, 0.2, 0.4, 0.6, 0.8, 1.0
Number of episode of source task	10000
Number of episode of target task	10000
Number of trial self-transfer	1
Number of trial transfer learning	3

B の行動が選択され，最短経路が変化せず 15 ステップで横ばいとなる学習曲線が得られたと考えられる．

6.4.3　考察

高い信頼率においては，4.3 節で指摘したように知識を結合しても，知識 B と知識 C のどちらかしか使用されない事を意味する．$\rho = 0.2$ においては，$\rho = 0$ の学習曲線より高いパフォーマンスで学習がスタートしている事がみてとれる．これは，知識の再利用による効果であり，ジャンプスタートと呼ばれる現象である．異なる解法を持つ知識であっても，信頼率を付加して再利用することにより，一から学習するより効率的に最適解の獲得が行え，さらに収束も速い．$\rho = 0.4, 0.6$ においては，知識 B の最短経路のステップ数から学習がスタートし，その後パフォーマンスの改善が行われ，フィールド A における最短経路への収束が見られる．この実験条件においては，学習初期段階では知識 B の

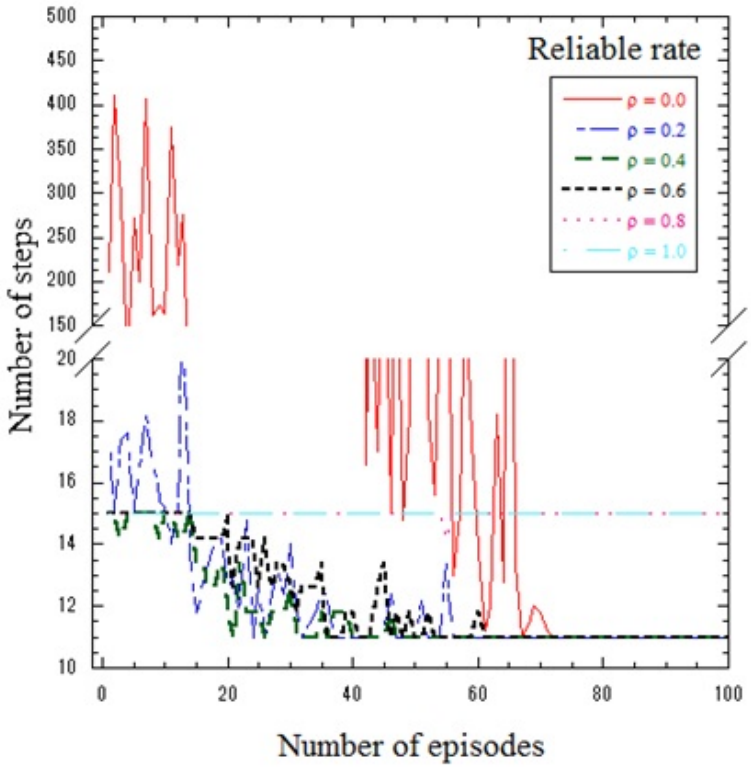


図 6.5 結合知識を用いた最短経路探査問題の学習曲線

最短経路を通過し，なおかつフィールド A における最短経路を探査していると考えられる．例として，表 6.2 に初期座標 (1, 1) における強化学習エージェントが獲得した各知識の行動価値を示す．フィールド B にて獲得した知識 B における行動価値は，図 6.4(b) の通り下方向への行動が高い．また知識 C においても同様に，行動価値は表 6.2 の通り右方向の行動価値が高い．$\rho = 1.0$ における結合した知識の行動選択確率を見ると，右方向への移動が 0.368%であり，下方向移動が 99.632%と極端な差が確認できる．これにより，エージェントは主に知識 B の行動を選択し，知識 B の最短経路しか通過しなかったと考えられる．これに対し，知識 B の最短経路から新たな最短経路への探査を見せた $\rho = 0.2$ においては，右方向と下方向移動の行動価値の差が少なくなり，右方向の行動も選択されやすくなっていると考えられる．これにより，フィールド A の新たな最短経路を探査できたと考えられる．

これらの結果から，信頼率 ρ を用いて再利用する知識の行動価値を調整する事により，新たな環境（類似タスク）における探査を促すことが可能である．さらに，過去に得られた最短経路を使いながら，新たな環境の最短経路を獲得できることが実験結果より示せた．

表 6.2 統合知識の行動価値と行動選択確率の例

Value \ Action	Move right	Move below	stop
Knowledge B	0.000	0.191	0.006
Knowledge C	0.163	0.000	0.008
Knowledge D ($\rho = 1.0$)	0.163	0.191	0.014
Select probablitiy	0.368 %	99.632 %	0.000 %
Knowledge D ($\rho = 0.2$)	0.033	0.038	0.003
Select probability	26.876%	73.057%	0.067%

6.5 結言

本章では，エージェントが強化学習により獲得した知識をクラウドを活用して知識の統合や保存をする知識処理に関する議論を行った．

多台数のエージェントが強化学習により知識を獲得し，膨大な量になると予想される知識を統合する手法の提案を行い，計算機実験から提案手法の有用性を確認した．これらの結果より，以下の成果が得られた．

(1) 提案手法により結合した知識は，目的は同一であるが解が異なる環境においても効果的に知識の再利用が可能であることを計算機実験により確認した．

(2) 知識の統合が可能になったことで，それぞれの環境でエージェントが獲得した知識を個別に保管することなく，統合による保管知識量の削減が可能になることが示唆された．

上記の成果より，多くの強化学習エージェントやロボットがクラウドに接続し，知識の保存・共有・再利用を単に広大なコンピューティングリソースに依存するだけでなく，効率的・効果的にクラウドを活用できることが示唆される．

第7章　結論

7.1　本論文の成果

MARS と MARL が融合したシステムの実世界応用には，エージェントらが学習し獲得した知識を共有・再利用するプロセスが重要であり，学習アルゴリズム単体では実世界のニーズには対応できない．学習アルゴリズムだけでなく，学習した知識を共有・再利用，他にも知識の獲得判定や，獲得知識の保存・処理など関連機能との統合が必要であると指摘した．そこで本論文では，ヘテロジーニアスを前提とした強化学習エージェントやロボットによる知識の共創を提案し，具体系として知識共創フレームワークを提案・検討した．知識共創フレームワークの要素技術として，本論文では以下の機能を提案し実験により有用性を確認した．

- ヘテロジーニアスな MARL における転移学習法
- 知識の再利用時における転移率調整法
- 強化学習における自律的収束推定法
- 外部計算機資源を用いた獲得知識の統合手法

以下，各章で得られた研究結果をまとめる．

第 3 章では，これまで深く議論がなされてこなかったヘテロジーニアスなエージェント間における転移学習法と，新たにエージェントがシステムに参加した場合における ITM の設計作業量削減を目的に，OITM を用いた階層的転移学習を提案した．本手法は，これまで個別に設計されていた ITM をオントロジにより統合することで，ITM の記述を一元管理している．これにより，新たなエージェントが OITM に接続されれば，既に接続されているエージェント群の知識が再利用可能である．自己転移や行動空間がヘテロジーニアス，状態空間がヘテロジーニアス，その双方がヘテロジーニアスという実験条件を用いた追跡問題による計算機実験から，全ての実験条件にてこれまでの転移学習と同様に JS が発現し，知識の再利用が可能であることを示した．さらに，知識の関数近似によるデータ量軽減化に関しても計算機実験を行い，特に，ANN における知識の関数近似は，階層的転移学習に悪影響を及ぼさないことを確認した．

第 4 章では，Taylor らの転移学習のように単純に知識を再利用すると負の転移が発現する可能性を指摘し，転移率を導入した知識の再利用法を提案した．高野らは既に転移率に相当するパラメータを提案していたが，常に再利用する知識がバイアスとして新たに学習する知識に影響を及ぼすため，本章の実験により転移は可能であるが最終的に獲得する知識に影響が現れるケースを再現した．そこで，本章ではエージェントが新たな環境で十分に学習した場合，再利用する知識の再利用度合を低下させる手法を提案した．最短経路問題や追跡問題を用いた計算機実験から既存手法で発現した，転移の効果が十分に得られない場合においても，提案手法により JS の発現や低い DCS が発現を確認した．

第 5 章では，これまで人間の目視や直感に依存して判断されていた強化学習の収束判定（知識の獲得）を，エージェント自身に行わせることを目的に，フラクタル次元解析を用いた収束推定法を提案した．提案手法として，効果の検証のために画像処理を用いた収束推定法である CEFDi と，エージェントへの実装を前提にした数値情報から直接フラクタル次元を計算する CEFDn を提案した．最短経路問題や追跡問題を用いた計算機実験により，学習の収束が推定可能であることが示唆され，また強化学習の学習曲線にフラクタル性が存在する可能性を実験的に示した．

第 6 章では，知識の獲得が目的である強化学習に対して，深く検討がなされてこなかった獲得知識の処理に関する議論を行った．エージェントが強化学習により獲得した知識を，クラウドを活用して知識の統合をする手法を提案した．本章における提案手法でも計算機実験により評価を行い，同目的異解法の関係にある知識を統合し，新たな環境でも再利用可能であることを確認した．

本研究では，以上の 4 つのアプローチで知識共創フレームワークの実現を目指し，強化学習エージェントやロボットによる知識共創の第一歩を実現した．本論文で提案した手法が全ての問題を解決するわけではない．しかし，本論文に得た結果，成果は，知識共創フレームワークの構築手法の 1 つとして有用な知見になると考えられる．

7.2 今後の展望

本節では，本論文で議論してきた提案手法に対し，まだ未検討であったり未解決な要素に関して議論を行う．知識共創フレームワークは，様々な要素技術が複雑に統合されるため，実世界応用のためには更なる議論・検討が必要であると考えられる．以下から示す課題に対して積極的な議論が望まれる．

7.2.1 行動空間と状態空間の標準化

本研究では，前提としてエージェントが車輪やクローラを用いた自律移動型のロボットとし，議論を行った．しかし，ロボットの身体性が類似関係にあっても，内部システムの構成は開発者に委ねられ，また強化学習アルゴリズムの種類によっても行動空間 A や状態空間 S の構成が異なることは想像に難くない．本研究の提案手法である OITM により，ある程度のヘテロジーニティを吸収できるとしても限界がある．今後は，全てのエージェントが OITM に接続できるような内部メカニズムを設計しなければならないと考えられる．

この課題は，エージェントの内部プログラムにおけるインタフェースの標準化であると言える．ハードウェアと制御ソフトウェアのインタフェースを接続するミドルウェアは，RT ミドルウェアや Robot opearating syste (ROS) など以前から活発に議論・開発されている [100,101]．ROS は，制御ソフトウェアや通信プロトコル，開発したプログラム共有など幅広い統合を可能にする強力なミドルウェアである．制御ソフトウェアとその上位に存在する知識を接続する OITM は，RT ミドルウェアと比較すれば上位層のインタフェースを統合する手法である．今後は，インタフェースであるエージェントの行動空間 A や状態空間 S の整合のために，RT ミドルウェアや ROS などのフレームワークの活用・統合も必要である．

本論文における実験のように，計算機実験においては行動空間 A と状態空間 S が離散状態で記述することが可能であった．C 項に示すような，実ロボットは基本的には行動空間と状態空間が連続状態で記述される．センサー分解能などの粗視化により，実ロボットの連続空間を離散状態として記述することも可能であるが，センシング精度や範囲などに悪影響を及ぼす可能性も考えられる．今後は，離散状態と連続状態を接続する方法論の議論も必要である．今後は，付録 C に示す著者らが開発した計算機内のエージェントとヘテロジーニティが低い実ロボットを用いて議論を行い，上述の問題解決を図る．

7.2.2 オントロジサーバの標準化

提案手法の 1 つである階層的転移学習は，これまで知識処理で活発に議論されてきたオントロジ工学を基にしている．しかし，活発に議論されてきたがゆえに様々なオントロジ構築フレームワークや方法論が乱立している状態であると言える．これらのフレームワークは，相互運用が行えるものは少なく本研究の OITM においても同様の問題が発生する．

本論文では，オントロジは溝口らのオントロジ工学 [65] を参考に構築を行った．ある 1 つのオントロジに特化すると，他のオントロジとの相互運用は難しく今後はオントロジ構築の標準化を行うか，各オントロジの相互運用性を高める議論が必要である．さらに，エージェントが利用可能なオントロジを公開・管理するサーバも必要となり，それらのシステムの標準化も必要となると考えられる．

7.2.3 実世界実タスクにおける実証実験

本論文での検証実験は，ほとんどが計算機実験によるものであった．実世界での運用を考慮すると，環境のダイナミクスは動的で複雑である．本論文で提案した手法は動的環境を前提としているが，今後は実環境における実機を用いた強化学習ロボットにおける検証実験と，その結果のフィードバックが不可欠である．

前提として設定した通信インフラ等の問題は，本課題に関連する事項となる．知識共創フレームワークは，強化学習エージェントやロボットがクラウドと接続可能となることで機能する．実際のインターネット環境を考慮すると，クラウドを構成するサーバ群との物理的な距離によるスループット低下や，Network interface card (NIC) などの性能，利用可能プロトコルにより知識データの送受信が低速となる可能性も十分に考えられる．知識データ自体は関数近似により容量を低減させることが出来るが，実際の環境にて実証する必要がある．さらに，クラウドとの通信プロトコルに関しては，7.2.1 項で述べたミドルウェアと同様に，既に議論が成されている RSNP [102] などの確立しつつあるフレームワークとの統合を検討する必要がある．

クラウドに関しては，付録 D に示す Hadoop などのフレームワークを用いて，知識の統合手法などの検証を行う必要がある．Hadoop は現在，クラウドサービスとして一般的に機能提供がなされているフレームワークであり，実際のクラウド環境を模擬できる．今後はクラウド模擬システムを用いて提案手法の検証を行い，実クラウドサービスへの知識共創フレームワークの実装を行う．

7.2.4 知識共創フレームワークの運用体制確立

知識共創フレームワークを実世界において運用する場合，様々なエージェントが必要であり，またそれらを運用する人々が関わる．もちろん，本論文で提案した手法を用いるだけでは即座に強化学習ロボットが実世界で使えるようになるとは限らない．

実世界運用においては，図 7.1 に示すような体制が必要であると考える．図 7.1 では，強化学習エージェントやロボットを開発・供給する会社が存在し，それらエージェントのための知識をシミュレーションにより生成する者がいる．さらには，実環境に適応できるように既存の知識を用いてロボットを訓練する場所も必要であると考える．そして，家庭や工事現場などで運用されるエージェントたちが存在し，それらの全てがクラウドを介して知識の共創を行うことで長期的な知識共創の運用が出来ると考える．

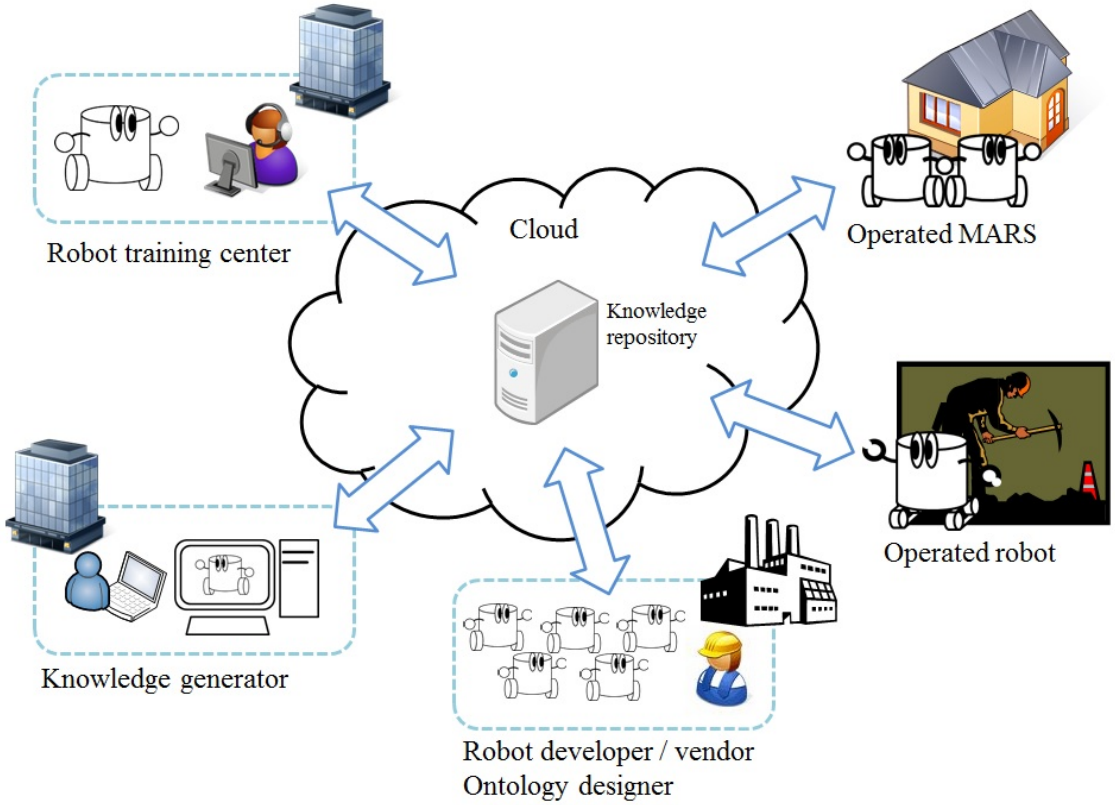


図 7.1 知識共創フレームワークの運用イメージ

参考文献

[1] 一般社団法人日本機械工業連合会. 平成 25 年度ロボット産業・技術の振興に関する調査報告書, 3 2014.

[2] 文部科学省. 科学技術基本計画, 8 2011.

[3] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura. The intelligent asimo: System overview and integration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, Vol. 3, pp. 2478–2483. IEEE, 2002.

[4] Takashi Uchiyama, Toshihiko Morita, and Naoyuki Sawasaki. Development of personal robot. In *Robotics Research*, pp. 319–336. Springer, 2011.

[5] Joseph L Jones. Robots at the tipping point: the road to irobot roomba. *Robotics & Automation Magazine, IEEE*, Vol. 13, No. 1, pp. 76–78, 2006.

[6] 村井亮介, 酒井龍雄, 上松弘幸, 中嶋久人, 三谷宏一, 北野斉. 自律移動ロボット群による搬送システムの実用化. 日本ロボット学会誌, Vol. 28, No. 3, pp. 311–318, 2010.

[7] 下笹洋一. 警備ロボットの現状. 映像情報メディア学会誌: 映像情報メディア, Vol. 57, No. 1, pp. 79–82, 2003.

[8] Hisayoshi Sugiyama, Tetsuo Tsujioka, and Masashi Murata. Coordination of rescue robots for real-time exploration over disaster areas. In *Object Oriented Real-Time Distributed Computing (ISORC) 2008 11th IEEE International Symposium on*, pp. 170–177. IEEE, 2008.

[9] Raffaello D'Andrea. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *Automation Science and Engineering, IEEE Transactions on*, Vol. 9, No. 4, pp. 638–639, 2012.

[10] Galia V Tzvetkova. Robonaut 2: Mission, technologies, perspectives. *Journal of Theoretical and Applied Mechanics*, Vol. 44, No. 1, pp. 97–102, 2014.

[11] Naoaki Yonezawa, Koshi Kashiwazaki, Kazuhiro Kosuge, Yasuhisa Hirata, Yusuke Sugahara, Mitsuru Endo, Takashi Kanbayashi, Koki Suzuki, Kazunori Murakami, and Kenichi Nakamura. Car transportation system grasping two drive wheels. In

Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 4086–4091. IEEE, 2012.

[12] Hisayoshi Sugiyama, Tetsuo Tsujioka, and Masashi Murata. Real-time exploration of a multi-robot rescue system in disaster areas. *Advanced Robotics*, Vol. 27, No. 17, pp. 1313–1323, 2013.

[13] Alessandro Marino, Lynne E Parker, Gianluca Antonelli, and Fabrizio Caccavale. A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling. *Journal of Intelligent & Robotic Systems*, Vol. 71, No. 3-4, pp. 423–444, 2013.

[14] Carlos Nieto-Granda, John G Rogers, and Henrik I Christensen. Coordination strategies for multi-robot exploration and mapping. *The International Journal of Robotics Research*, p. 0278364913515309, 2014.

[15] 浅田稔. 強化学習の実ロボットへの応用とその課題. 人工知能学会誌, Vol. 12, No. 6, pp. 831–836, 1997.

[16] 大倉和博, 保田俊行, 松村嘉之. 構造進化型人工神経回路網による swarm robotics のための適応的協調行動の生成. 日本機械学会論文集 C 編, Vol. 77, No. 775, pp. 966–979, 2011.

[17] 谷口忠大, 椹木哲夫. 双シェマモデル: 自律エージェントの為の自己組織化機械学習手法の提案. 人工知能学会論文誌, Vol. 19, pp. 493–501, 2004.

[18] 大内東, 山本雅人, 川村秀憲. マルチエージェントシステムの基礎と応用: 複雑系工学の計算パラダイム. コロナ社, 2002.

[19] Erfu Yang and Dongbing Gu. A survey on multiagent reinforcement learning towards multi-robot systems. In *Computational Intelligence and Games (CIG) 2005 IEEE Symposium on*, 2005.

[20] Manuel Graña, Borja Fernandez-Gauna, and Jose Manuel Lopez-Guede. Cooperative multi-agent reinforcement learning for multi-component robotic systems: guidelines for future research. *Paladyn*, Vol. 2, No. 2, pp. 71–81, 2011.

[21] Sachiyo Arai, Katia Sycara, and Terry R Payne. Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain. In *PRICAI 2000 Topics in Artificial Intelligence*, pp. 125–135. Springer, 2000.

[22] Sachiyo Arai, Katia Sycara, and Terry R Payne. Multi-agent reinforcement learning for planning and scheduling multiple goals. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pp. 359–360. IEEE, 2000.

[23] 荒井幸代. マルチエージェント強化学習: 実用化に向けての課題・理論・諸技術との融合. 人工知能学会誌, Vol. 16, No. 4, pp. 476–481, 2001.

[24] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, Vol. 337. Amherst, MA, 1993.

[25] Maja J Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, Vol. 4, No. 1, pp. 73–83, 1997.

[26] 高玉圭樹. マルチエージェント学習 – 相互作用の謎に迫る –. コロナ社, 2003.

[27] 保田俊行. 強化学習によるマルチロボットシステムの協調行動獲得に関する研究. PhD thesis, 神戸大学, 2006.

[28] 福田敏男, 船戸大輔, 新井史人. 多重強化学習法に基づく群ロボットシステムにおける環境変化認識. 日本機械学會論文集. C 編, Vol. 66, No. 643, pp. 864–869, 2000.

[29] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[30] David H Wolpert and William G Macready. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[31] Majid Nili Ahmadabadi, Masoud Asadpour, and Eiji Nakano. Cooperative q-learning: the knowledge sharing issue. *Advanced Robotics*, Vol. 15, No. 8, pp. 815–832, 2001.

[32] Majid Nili Ahmadabadi and Masoud Asadpour. Expertness based cooperative q-learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, Vol. 32, No. 1, pp. 66–76, 2002.

[33] 保田俊行, 大倉和博. 確率ネットワークを用いた強化学習ロボットの獲得戦略の保存と利用 (機械力学, 計測, 自動制御). 日本機械学會論文集. C 編, Vol. 73, No. 736, pp. 3212–3219, 2007.

[34] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. Cloud robotics: architecture, challenges and applications. *Network, IEEE*, Vol. 26, No. 3, pp. 21–28, 2012.

[35] Oliver Zweigle, René van de Molengraft, Raffaello d'Andrea, and Kai Häussermann. Roboearth: connecting robots worldwide. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pp. 184–191. ACM, 2009.

[36] Markus. Waibel, Michael. Beetz, Javier. Civera, Raffaello. D'Andrea, Jos. Elfring, Dorian. Galván-López, Kai. Haüssermann, Rob. Janssen, J. M M Montiel, Alexander Perzylo, Björn. Schießle, M. Tenorth, Oliver. Zweigle, and René. van de Molengraft. A world wide web for robots roboearth. *Robotics & Automation Magazine, IEEE*, Vol. 18, No. 2, pp. 69–82, June 2011.

[37] Dominique Hunziker, Mohanarajah Gajamohan, Markus Waibel, and Raffaello D'Andrea. Rapyuta: The roboearth cloud engine. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 438–444. IEEE, 2013.

[38] 松井藤五郎, 犬塚信博, 世木博久, 伊藤英則. 強化学習結果の再構築への概念学習の適用 using concept. 人工知能学会論文誌, Vol. 17, No. 2, pp. 135–144, 2002.

[39] 松井藤五郎. 自律型エージェントの行動学習に関する研究. PhD thesis, 名古屋工業大学, 2003.

[40] Matthew E Taylor. *Transfer in Reinforcement Learning Domains*, Vol. 216. Springer, 2009.

[41] Georgios Boutsioukis, Ioannis Partalas, and Ioannis Vlahavas. Transfer learning in multi-agent reinforcement learning domains. In *Recent Advances in Reinforcement Learning*, pp. 249–260. Springer, 2012.

[42] Adam Taylor, Ivana Dusparic, Edgar Galván-López, Siobhán Clarke, and Vinny Cahill. Transfer learning in multi-agent systems through parallel transfer. In *Workshop on Theoretically Grounded Transfer Learning at the 30th International Conference on Machine Learning (Poster)*, Vol. 28, 2013.

[43] 上田完次, 黒田あゆみ. 共創とは何か. 培風館, 2004.

[44] 山田和明, 中小路久美代, 山本恭裕. オンラインコミュニティにおける知識共創のモデル. 第 4 回知識流通ネットワーク研究会, 2009.

[45] TD Sofianti, K Suryadi, R Govindaraju, and B Prihartono. Customer knowledge co-creation process in new product development. In *Proceedings of the World Congress on Engineering*, Vol. 1, 2010.

[46] 金野武司, 森田純哉, 橋本敬. コミュニケーションシステムの形成過程に見る知識共創の基盤. 第 3 回知識共創フォーラム予稿集, Vol. 3, pp. 8–1, 2013.

[47] Rolf Pfeifer, Christian Scheier, 石黒章夫, 小林宏, 細田耕. 知の創成—身体性認知科学への招待—. 共立出版, 2001.

[48] 山下雅史. 自律分散ロボットにおける共有知識の創発. 計測と制御= Journal of the Society of Instrument and Control Engineers, Vol. 38, No. 10, pp. 654–657, 1999.

[49] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, Vol. 8, No. 3-4, pp. 279–292, 1992.

[50] 荒井幸代, 宮崎和光, 小林重信. マルチエージェント強化学習の方法論: Q-learning と profit sharing による接近. 人工知能学会誌, Vol. 13, No. 4, pp. 609–618, 1998.

[51] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, Vol. 10, pp. 1633–1685, 2009.

[52] 神嶌敏弘. 転移学習. 人工知能学会誌, Vol. 25, No. 4, pp. 572–580, 2010.

[53] Matthew E Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, pp. 879–886. ACM, 2007.

[54] Samuel Barrett, Matthew E. Taylor, and Peter Stone. Transfer learning for reinforcement learning on a physical robot. In *Ninth International Conference on Autonomous Agents and Multiagent Systems - Adaptive Learning Agents Workshop (AAMAS - ALA)*, 2010.

[55] 高野敏明. 同一エージェント間における転移学習を用いた学習の高速化に関する研究. PhD thesis, 三重大学, 2012.

[56] Luiz A. Celiberto Jr., Jackson P. Matsuura, Ramón. López de Mántaras, and Reinaldo AC. Bianchi. Using transfer learning to speed-up reinforcement learning: A cased-based approach. In *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American*, pp. 55–60, Oct 2010.

[57] Balaji. Lakshmanan and Ravindran. Balaraman. Transfer learning across heterogeneous robots with action sequence mapping. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 3251–3256, Oct 2010.

[58] Bócsi. Bocsi, Lehal. Csató, and Jan. Peters. Alignment-based transfer learning for robot models. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–7, Aug 2013.

[59] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 22, No. 10, pp. 1345–1359, 2010.

[60] Adam Taylor, E Galván-López, S Clarke, and V Cahill. Accelerating learning in multi-objective systems through transfer learning. In *In a Special Session on Learning and Optimization in Multi-Criteria Dynamic and Uncertain Environments at the International Joint Conference on Neural Network*, 2014.

[61] Fernando Fernández, Javier García, and Manuela Veloso. Probabilistic policy reuse for inter-task transfer learning. *Robotics and Autonomous Systems*, Vol. 58, No. 7, pp. 866–871, 2010.

[62] Peter Vrancx, Yann-Michaël De Hauwere, and Ann Nowé. Transfer learning for multi-agent coordination. In *ICAART (2)*, pp. 263–272, 2011.

[63] 小林祐一. 強化学習のための自律分散型関数近似法. PhD thesis, 東京大学, 2001.

[64] 赤間世紀. オントロジーがわかる本. 工学社 I/O BOOKS, 2010.

[65] 溝口理一郎. オントロジー研究の基礎と応用. 人工知能学会誌, Vol. 14, No. 6, pp. 977–988, 1999.

[66] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, Vol. 5, No. 2, pp. 199–220, 1993.

[67] 古崎晃司, 溝口理一郎, 同上. オントロジー構築ツールの現状. 人工知能学会誌, Vol. 20, No. 6, pp. 707–714, 2005.

[68] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: Implementing the semantic web recommendations. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, WWW Alt. '04, pp. 74–83, New York, NY, USA, 2004. ACM.

[69] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Software-Practice and Experience*, Vol. 31, No. 2, pp. 103–128, 2001.

[70] 伊藤英毅. オントロジーを利用した知識の共有/再利用. *Unisys Technology Review*, No. 64, 2000.

[71] 中山裕司. オントロジーを用いた異機種ロボットにおける情報共有機構の構築. Master's thesis, 奈良先端科学技術大学院大学, 2006.

[72] Les Gasser. Representing and using organizational knowledge in distributed ai systems. *Distributed Artificial Intelligence*, pp. 55–78, 1989.

[73] Steffen Nissen. Neural networks made simple. *Software 2.0*, Vol. 2, pp. 14–19, 2005.

[74] Steffen Nissen. Large scale reinforcement learning using q-sarsa(λ) and cascading neural networks. Master's thesis, 2007.

[75] Toshiaki Takano, Haruhiko Takase, Hiroharu Kawanaka, and Shinji Tsuruoka. Effective reuse method for transfer learning in actor-critic. In *SCIS & ISIS*, Vol. 2010, pp. 137–141. 日本知能情報ファジィ学会, 2010.

[76] Toshiaki Takano, Haruhiko Takase, Hiroharu Kawanaka, and Shinji TSURUOKA. Accelerate learning processes by avoiding inappropriate rules in transfer learning for actor-critic. In *Proceedings of the Second International Workshop on Regional Innovation Studies:(IWRIS2010)*, No. 2, pp. 55–58. Graduate School of Regional Innovation Studies, Mie University, 2011.

[77] Toshiaki Takano, Haruhiko Takase, Hiroharu Kawanaka, Hidehiko Kita, Terumine Hayashi, and Shinji Tsuruoka. Transfer learning based on forbidden rule set in actor-critic method. *International Journal of Innovative Computing, Information and Control*, Vol. 7, No. 5 B, pp. 2907–2917, 2011.

[78] 明彦山口, 淳高松, 司小笠原. 強化学習によるロボットの動作獲得のための基底関数に基づく行動空間生成手法 dcob : 実機多自由度ロボットの匍匐動作への適用. 第 2010 巻, pp. 2P1–G10(1)–2P1–G10(4). 一般社団法人日本機械学会, 2010.

[79] 鯉江康弘, 山本博巳, 山地憲治. マルチエージェント手法による co2 排出権市場の分析. *Journal of Japan Society of Energy and Resources*, Vol. 32, No. 2, pp. 1–8, 2011.

[80] Kan Sichao, Hiromi Yamamoto, and Kenji Yamaji. Evaluation of co2 free electricity trading market in japan by multi-agent simulations. *Energy Policy*, Vol. 38, No. 7, pp. 3309–3319, 2010.

[81] 池上渉一, 浅田稔, 細田耕. 非同期政策更新に基づくマルチエージェント同時強化学習による協調行動の獲得. 日本ロボット学会創立 20 周年学術講演会, p. 3H3, 2002.

[82] 伊藤昭, 金渕満. 知覚情報の粗視化によるマルチエージェント強化学習の高速化-ハンターゲームを例に. 電子情報通信学会論文誌 D, Vol. 84, No. 3, pp. 285–293, 2001.

[83] 大谷雅之, 佐藤寛之, 服部聖彦, 高玉圭樹. 複数ロボット協調による大規模構造物組み立てにおける故障ロボット回収の影響. 電気学会論文誌 C (電子・情報・システム部門誌), Vol. 133, No. 9, pp. 1729–1737, 2013.

[84] Benoit B Mandelbrot. How long is the coast of britain. *Science*, Vol. 156, No. 3775, pp. 636–638, 1967.

[85] 松尾崇, 仲久保正人, 山本圭治郎. 自然界のフラクタルパターンと感性 (感性の評価)(シンポジウム: 福祉工学). 機械力学・計測制御講演論文集, 第 2001 巻, p. 326. 一般社団法人日本機械学会, 2001.

[86] 大野博之, 小島圭二. 岩盤割れ目のフラクタル (その 2): フラクタル特性と分布のばらつき. 応用地質, Vol. 34, No. 2, pp. 58–72, 1993.

[87] 佐藤明人. 大腸上皮性腫瘍腺口形態 (pit pattern) のフラクタル解析: pit pattern の定量評価と病理組織診断との対比. 新潟医学会雑誌, Vol. 119, No. 8, pp. 464–473, 2005.

[88] Tian-Pau Chang, Hong-Hsi Ko, Feng-Jiao Liu, Pai-Hsun Chen, Ying-Pin Chang, Ying-Hsin Liang, Horng-Yuan Jang, Tsung-Chi Lin, and Yi-Hwa Chen. Fractal dimension of wind speed time series. *Applied Energy*, Vol. 93, pp. 742–749, 2012.

[89] 横山秀史, 永田茂, 片山恒雄. 研究速報: フラクタル次元を用いた人間行動の定量的分析. 生産研究, Vol. 43, No. 12, pp. 41–44, 1991.

[90] 熊谷善彰. 円ドルレートティックデータの週次フラクタル次元. *Journal of the Operations Research Society of Ja*, Vol. 45, No. 4, pp. 457–470, dec 2002.

[91] 山際謙太. フラクタル解析を用いた金属破断面の特性化と金属破壊機構解明への応用. Master's thesis, 2000.

[92] 巽二郎. 根の形態測定法 : フラクタル解析の利用 (作物の形態研究法 : マクロからミクロまで, 連載ミニレビュー). 日本作物學會紀事, Vol. 76, No. 4, pp. 604–609, oct 2007.

[93] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. Nih image to imagej: 25 years of image analysis. *Nature methods*, Vol. 9, No. 7, pp. 671–675, 2012.

[94] 本田勝也. フラクタル （シリーズ＜非線形科学入門＞1）. 朝倉書店, 2002.

[95] 中川匡弘. カオス・フラクタル感性情報工学. 日刊工業新聞社, 2010.

[96] 秀明蝶名林, 宏典根來, 宏友大内. フラクタル次元解析を用いた景観認知による可視化モデルの複雑性の定量化手法. 日本建築学会技術報告集, Vol. 22, pp. 549–552, dec 2005.

[97] 山際謙太, 酒井信介, 横堀壽光. フラクタル解析を用いた tial 高温域破壊機構の特性化. 日本材料強度学会誌, Vol. 35, No. 3, pp. 53–60, 2001.

[98] 内部英治, 浅田稔, 野田彰一, 細田耕. 視覚に基づく強化学習による移動ロボットの多重タスクの達成. 日本ロボット学会学術講演会予稿集, pp. 609–610, 1994.

[99] R Matthew Kretchmar. Parallel reinforcement learning. In *The 6th World Conference on Systemics, Cybernetics, and Informatics*. Citeseer, 2002.

[100] Noriaki Ando, Takashi Suehiro, Kosei Kitagaki, Tetsuo Kotoku, and Woo-Keun Yoon. Rt-middleware: distributed component middleware for rt (robot technology). In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3933–3938. IEEE, 2005.

[101] Steve Cousins, Brian Gerkey, Ken Conley, and Willow Garage. Sharing software with ros [ros topics]. *Robotics & Automation Magazine, IEEE*, Vol. 17, No. 2, pp. 12–14, 2010.

[102] 陽介土屋, 昭二鈴木, 由花加藤, 雅彦成田. Rsnp を利用したロボットサービスコンテスト (ビッグデータで変わる環境知能とロボット, 及びその他一般). 電子情報通信学会技術研究報告. CNR, クラウドネットワークロボット, Vol. 112, No. 233, pp. 37–40, oct 2012.

[103] R. W. Stineman. A consistently well behaved method of interpolation. *Creative Computing*, Vol. 6, No. 7, pp. 54–57, 1980.

[104] 太田一樹, 岩崎正剛, 猿田浩輔, 下垣徹, 藤井達朗, 山下真一, 濱野賢一朗. Hadoop 徹底入門: オープンソース分散処理環境の構築 第 2 版. 翔泳社, 2013.

謝辞

本研究は 2012 年度から実施されました．2014 年度から，本研究の一部は東京電機大学総合研究所研究 Q14-J01 の助成を受けたものです．ここに感謝の意を表します．

本研究の遂行ならびに論文の作成にあたり，親切な御指導賜りました東京電機大学工学部情報通信工学科 鈴木剛 教授に謹んで深謝の意を表します．いつも思い付きで行動し突き進む私を，鈴木教授は辛抱強く見守ってくださいました．鈴木教授は，私が学部 4 年の時から指導教員として面倒見てくださり，週末は一緒に酒を酌み交わし，何か失敗した時も温かく励ましていただきました．鈴木教授がいなければ，私はここまで来ることが出来ませんでした．今後とも御指導御鞭撻の程よろしくお願い申し上げます．

本論文をまとめるにあたり，ご多用中にもかかわらず有益な御助言と御教示を賜りました，東京電機大学工学部情報通信工学科 月本洋 教授，東京電機大学未来科学部情報メディア学科 中島克人 教授，東京電機大学工学部電気電子工学科 五十嵐洋 准教授，に心より感謝の意を表します．

産業技術総合研究所知能システム研究部門フィールドロボティクス研究グループ 神村明哉 主任研究員，産業技術総合研究所知能システム研究部門スマートモビリティ研究グループ 富田康治 主任研究員に感謝申し上げます．神村博士の御助言は，常に良い刺激となり，いつも親身になって相談に乗っていただきました．また，サーボ基板など多くの技術を提供していただき，ロボット開発に多大なる貢献を賜りました．ご多用のところ，博士論文の副査もなっていただきました．私の研究に多大なるご協力をいただき，深く感謝いたします．富田博士には，いつも有益なご助言や論文の添削をして下さいました．学会など様々な場面で助けていただきました．心より感謝いたします．

東京電機大学工学部情報通信工学科 定松宣義 講師，順天堂大学大学院医学研究科研究基盤センター生体工学研究部門 江原義郎 先任准教授（当時）には大学 1 年からお世話になりました．私は学部生時代，学生職員として定松先生の下で仕事させていただきました．院生になってからも，副手として採用していただき大変お世話になりました．江原先生は，博士後期課程入学前から私に様々な御助言を下さいました．博士後期課程に入学してからは，一緒にお酒を飲み煙草を吸い，多くの励ましを頂きました．お二人の先生に改めて感謝いたします．

東京電機大学大学院工学研究科 澤井圭 助教は，私が学部 4 年の時から研究の直属の先輩であり，いつも親身になって相談に乗っていただきました．澤井助教は，民間企業を退職して博士号を取得されたこともあり，私の良き先輩良き見本でした．学会に行くといつもトラブルに見舞われる私を，いつも助けてくれました．ありがとうございました．

神奈川県立磯子工業高等学校（当時）の 佐々木淳 先生にも感謝いたします．佐々木先生からは，機械加工や電子工作の技術を沢山学ばせてていただきました．佐々木先生がいなければ，私の持つ技術やセンスは無かったと思います．また，私が大学へ進学する切っ掛けを作っていただいたのも佐々木先生でした．ありがとうございました．

東京電機大学工学部情報通信工学科 ネットワークロボティクス研究室の皆様に感謝いたします．特に，M1 の 田代淳史 氏と 村田雄太 氏には，作業量の膨大な私の研究テーマに協力していただきました．最後まで諦めずに私の研究に付いて来てくれました．小型全方向移動ロボットのプロトタイプを設計・開発してくれた，当時 B4 の 桑原由里 女史（旧姓）にも感謝いたします．皆様ありがとうございました．他にも，いつも呑みに行った暗号方式・暗号プロトコル研究室 M2 押切徹 氏，実験を手伝ってくれた情報通信工学科 1 年 藤原大地 氏，大月みなみ 女史にも感謝いたします．

I would like to express my gratitude to Senior Scientist Ronnie Johansson at Swedish Defence Research Agency (Totalförsvarets forskningsinstitut, FOI). Dr. Johansson gives me constructive comments and warm encouragement. I had a good time doing shopping of junk parts at Akihabara with Dr. Johansson.

株式会社富士通総研，一橋大学大学院社会学研究科地球社会研究専攻 朝倉隆道 氏に感謝します．博士課程の友人が少ない私にとって，会社の同期であり同じ博士を志す朝倉氏との議論や雑談は，とても楽しい時間でした．ありがとうございました．

富士通株式会社 村松殿，村田殿，石井殿にも感謝いたします．私がまだ社員の頃，石井殿は，教育係として時に厳しく，時に優しく指導していただきました．もう会議では居眠りしません．村松殿は仕事の相談に乗っていただいたり，私が富士通を辞める際に応援の御言葉を下さいました．その御言葉に大変感銘を受けました．村田殿には，オンサイトの仕事術やタバコミュニケーションの底力を教えていただきました．会社でお世話になった人をここで全て書くことはできませんが，皆さんには感謝いたします．

最後になりましたが，両親に感謝いたします．勉強嫌いであった私をここまで育てていただき，会社を退職し博士後期課程に入学した私を暖かく見守っていただきました．金銭的にも支援していただきました．これからは親孝行に努めたいと思います．

2015 年 3 月 河野 仁

研究業績

学術論文　（査読有）

[1] **河野仁**, 村田雄太, 神村明哉, 富田康治, 鈴木剛, “ヘテロジーニアスな複数台自律エージェントにおける階層的転移学習”, 計測自動制御学会論文集. （再投稿中）

[2] **Hitoshi Kono**, Akiya Kamimura, Kohji Tomita, Yuta Murata and Tsuyoshi Suzuki, “Transfer Learning Method Using Ontology for Heterogeneous Multi-agent Reinforcement Learning”, *International Journal of Advanced Computer Science and Application* (IJACSA), Vol. 5, No.10, pp. 156–164, 2014.

[3] Tsuyoshi Suzuki, Kazuki Kato, Emi Makihara, Takafumi Kobayashi, **Hitoshi Kono**, Kei Sawai, Kuniaki Kawabata, Fumiaki Takemura, Naoko Isomura and Hideyuki Yamashiro, “Development of Underwater Monitoring Wireless Sensor Network to Support Coral Reef Observation”, *International Journal of Distributed Sensor Networks*, Vol. 2014, Article ID 189643, doi:10.1155/2014/189643, pp.1–10, 2014.

[4] Kei Sawai, Shigeaki Tanabe, **Hitoshi Kono**, Yuta Koike, Ryuta Kunimoto and Tsuyoshi Suzuki, “Construction Strategy of Wireless Sensor Networks with Throughput Stability by Using Mobile Robot”, *International Journal of Advanced Computer Science and Applications* (IJACSA), Vol.5, No.2, pp.15–20, 2014.

[5] Kei Sawai, Shigeaki Tanabe, **Hitoshi Kono**, Tsuyoshi Suzuki and Kuniaki Kawabata, “Design and development of impact-resistant sensor node for launch deployment into closed area”, *Sensor Review*, Vol. 32 Issue 4 pp. 318–326, 2012

[6] 澤井圭, **河野仁**, 鈴木剛, 羽田靖史, 川端邦明, “投射配置による落下衝撃を考慮した無線センサ端末の耐衝撃機構の開発. 設計工学”, Vol.45, No.1, pp13–19, 2010.

書籍　（Book chapter, 査読有）

[7] **Hitoshi Kono**, Yuta Murata, Akiya Kamimura, Kohji Tomita and Tsuyoshi Suzuki, “Knowledge Co-creation Framework: Novel Transfer Learning method in Heterogeneous Multi-agent Systems”,*Distributed Autonomous Robotic Systems Springer Tracts in Advanced Robotics* (DARS STAR Volume), Springer Berlin Heidelberg. (in Press)

[8] Tsuyoshi Suzuki, Kei Sawai, **Hitoshi Kono** and Shigeaki Tanabe, “Sensor Network Deployment by Dropping and Throwing Sensor Nodes to Gather Information in Underground Spaces in a Post-Disaster Environment”, *Advances in Robotics - Modeling, Control and Applications*, iConcept Press, pp.303-321, ISBN: 978-1-461108-443, 2013.

国際会議　（査読有）

[9] **Hitoshi Kono**, Yuta Murata, Akiya Kamimura, Kohji Tomita and Tsuyoshi Suzuki, “Knowledge Co-creation Framework: Novel Transfer Learning method in Heterogeneous Multi-agent Systems”,*Proceedings of the International Symposium on Distributed Autonomous Robotic Systems* (DARS2014), pp.465–478, Daejeon Korea, November 2014.

[10] **Hitoshi Kono**, Kei Sawai and Tsuyoshi Suzuki, “Convergence Estimation Utilizing Fractal Dimensional Analysis for Reinforcement Learning”, *Proceedings of the SICE Annual Conference 2013*, pp.2752–2757, Nagoya Japan, September 2013.

[11] Kei Sawai, Tsuyoshi Suzuki, **Hitoshi Kono**, Yasushi Hada and Kuniaki Kawabata, “Development of a Sensor Node with Impact-Resistance Capability to Gather the Disaster Area Information”, *2008 Proceedings of the International Symposium on Nonlinear Theory and its Applications* (NOLTA’08), pp.17–20, Budapest Hungary, September 2008.

[12] Kei Sawai, **Hitoshi Kono**, Tsuyoshi Suzuki, Yasushi Hada and Kuniaki Kawabata, “Development of a Sensor Node with Impact Resistance Capability for Gathering Disaster Area Information”, *Proceedings of the 5th International Conference on Networked Sensing Systems* (INSS2008), p259, Kanazawa Japan, June 2008.

国内学会　（査読有）

[13] **河野仁**, 田代淳史, 神村明哉, 富田康治, 鈴木剛, “ヘテロジーニアスなエージェントを用いた転移学習のための転移率の効果”, 第 20 回ロボティクスシンポジア, 3 月 2015. （採択済）

[14] **河野仁**, 澤井圭, 鈴木剛, “知識の抽象化と階層化による自律ロボットの階層的転移学習”, 第 19 回ロボティクスシンポジア, pp.479–484, 3 月 2014.

[15] 澤井圭, **河野仁**, 鈴木剛, 羽田靖史, 川端邦明, “投射配置による落下衝撃を考慮した耐衝撃機構を備えた無線センサノードの開発”, 第 14 回ロボティクスシンポジア, pp.459–464, 3 月 2009.

国内学会　（査読無）

[16] **河野仁**, 村田雄太, 神村明哉, 富田康治, 鈴木剛, “クラウドを活用した強化学習エージェントの獲得知識統合手法の検討”, 第 15 回 計測自動制御学会 システムインテグレーション部門講演会, pp.2460–2463, 12 月 2014.

[17] 田代淳史, **河野仁**, 鈴木剛, “ヘテロジーニアスなマルチロボット転移学習における転移率の検討”, 第 30 回ファジィシステムシンポジウム, pp.620–623, 9 月 2014.

[18] **河野仁**, 村田雄太, 神村明哉, 富田康治, 鈴木剛, “知識の抽象化と階層化による複数台自律エージェントの階層的転移学習とその効果”, ロボティクス・メカトロニクス講演会 2014, 1A1-L02, 5 月 2014.

[19] 村田雄太, **河野仁**, 神村明哉, 富田康治, 鈴木剛, “フラクタル次元解析を用いた強化学習の収束推定法の開発”, ロボティクス・メカトロニクス講演会 2014, 1A1-K05, 5 月 2014.

[20] 齋藤敬, **河野仁**, 鈴木剛, “モーションキャプチャを用いた顧客動線情報収集手法の開発”, ロボティクス・メカトロニクス講演会 2014, 3P1-X02, 5 月 2014.

[21] **河野仁**, 鈴木剛, “強化学習におけるフラクタル次元解析を用いた学習収束推定法の検討”, ロボティクス・メカトロニクス講演会 2013, 1A2-M02, 5 月 2013.

[22] 金子智大, **河野仁**, 鈴木剛, “ロボカップサッカーにおける強化学習を用いたポジションの推定”, ロボティクス・メカトロニクス講演会 2013, 1P1-A08, 5 月 2013.

[23] 篠宮佑太, **河野仁**, 金子智大, 鈴木剛, “ロボカップサッカーにおける妨害行動を考慮したパスシステムの開発”, 第 28 回ファジィシステムシンポジウム, pp.1139–1142, 9 月 2012.

[24] **河野仁**, 熊倉靖裕, 篠宮佑太, 鈴木剛, “ロボカップサッカーにおける数的不利時の

マルチロボット協調ディフェンスシステム –ディレイシステムの開発– ”, ロボティクス・メカトロニクス講演会 2012, 2A2-Q08, 5 月 2012.

[25] 柴田龍一, **河野仁**, 鈴木剛, “高次局所自己相関関数を用いた画像評価による自己位置認識の開発”, ロボティクス・メカトロニクス講演会 2012, 2A2-Q07, 5 月 2012.

[26] 篠宮佑太, 鈴木剛, **河野仁**, “ロボカップサッカーにおけるトラップ機構の開発”, ロボティクス・メカトロニクス講演会 2011, 2A2-F07, 5 月 2011.

[27] **河野仁**, 澤井圭, 鈴木剛, “移動ロボットによる無線情報収集端末投射配置を実現する投射機構の開発と実装”, ロボティクス・メカトロニクス講演会 2010, 1A2-C10, 5 月 2010.

[28] 角田康, **河野仁**, 犬塚裕貴, 塚本正士, 鈴木剛, “ロボカップサッカーにおける協調的パスプレイを考慮したパサー動作のモデル化”, ロボティクス・メカトロニクス講演会 2010, 1P1-F27, 5 月 2010.

[29] 塚本正士, 角田康, **河野仁**, 鈴木剛, “ロボカップサッカーにおける協調的な相手プレイヤの位置推定”, ロボティクス・メカトロニクス講演会 2010, 1P1-F28, 5 月 2010.

[30] 犬塚裕貴, **河野仁**, 角田康, 鈴木剛, “ぼかし画像を用いたボール認識手法の提案”, ロボティクス・メカトロニクス講演会 2010, 1P1-F29, 5 月 2010.

[31] 澤井圭, 君塚祐司, 山下也, **河野仁**, 鈴木剛, “耐衝撃機構を備えた無線センサノードの無線通信特性の評価”, ロボティクス・メカトロニクス講演会 2009, 1A2-H07, 5 月 2009.

[32] **河野仁**, 澤井圭, 鈴木剛, 羽田靖史, 川端邦明, “投射配置による落下衝撃を考慮した耐衝撃機構を備えたセンサノードの開発”, 第 9 回システムインテグレーション部門講演会 (SI2008), pp.21–22, 12 月 2008.

[33] **河野仁**, 澤井圭, 川端邦明, 羽田靖史, 鈴木剛, “耐衝撃性を考慮した被災地情報収集センサノードの開発”, ロボティクス・メカトロニクス講演会 2008, 2P1-A10, 5 月 2008.

[34] 澤井圭, **河野仁**, 川端邦明, 羽田靖史, 鈴木剛, “レーザポインタの直線性を考慮したセンサノードの位置推定”, ロボティクス・メカトロニクス講演会 2008, 1A1-F15, 5 月 2008.

[35] **河野仁**, 澤井圭, 川端邦明, 羽田靖史, 鈴木剛, “耐衝撃性を考慮した被災地情報収集センサノードの開発”, 平成 19 年度電子情報通信学会東京支部学生会研究発表会, p.26, 3 月 2008.

展示会等

[1] あだちメッセ 2012, 主催： 東京商工会議所足立支部, 開催期間： 2012 年 11 月 2 日–3 日. （東京電機大学 ネットワークロボティクス研究室として出展）

[2] 第 25 回先端技術見本市 テクノトランスファー in かわさき 2012, 主催： 公益財団法人神奈川産業振興センター, 開催期間： 2012 年 7 月 11 日–13 日. （東京電機大学 ネットワークロボティクス研究室として出展）

[3] 2009 国際ロボット展, 主催： 日本ロボット工業会, 日刊工業新聞, 開催期間： 2009 年 11 月 25 日–28 日. （東京電機大学 ネットワークロボティクス研究室，理化学研究所，情報通信研究機構と共同出展）

[4] 第 22 回 先端技術見本市テクノトランスファー in かわさき 2009, 主催： 公益財団法人神奈川産業振興センター, 開催期間： 2009 年 7 月 8 日–10 日. （東京電機大学 ネットワークロボティクス研究室として出展）

[5] 2007 国際ロボット展, 主催： 日本ロボット工業会, 日刊工業新聞, 開催期間： 2007 年 11 月 28 日–12 月 1 日. （東京電機大学 ネットワークロボティクス研究室，理化学研究所，情報通信研究機構と共同出展）

受賞

[1] SI2014 優秀講演賞, 受賞者： 東京電機大学 河野仁, 村田雄太, 産業技術総合研究所 神村明哉, 富田康治, 東京電機大学 鈴木剛, 受賞日： 2014 年 12 月 17 日.

特許

[1] 出願名称： オムニホイール, 発明者： 鈴木剛, 河野仁, 特許出願人： 学校法人東京電機大学, 特許出願番号： 特願 2014-265085（出願日：平成 26 年 12 月 26 日）.

付 録A　マルチエージェント強化学習シミュレータ

本シミュレータは，数値計算によりマルチエージェント強化学習の追跡問題を実行するために，著者が開発したものである．最大 3 台までのハンターを用いて追跡問題を実行可能であり，全てのエージェントの行動履歴が保存可能である．任意のグリッドワールドの大きさ，エージェントの初期座標なども指定可能である．また，エージェントが獲得した知識を保存し，転移学習へ知識ファイルの継承が可能である．さらに，学習曲線のデータもリアルタイムにロギングされる．本シミュレータの仕様を以下に示す．

- 開発言語：C 言語
- 動作オペレーティングシステム：Linux (Scientific Linux 6.5)
- プラットフォーム：Lenovo Think Station E30, Lenovo Think Server TS140
- 最大エージェント数：ハンター× 3, 獲物× ∞
- グリッドワールド内の任意の位置に障害物（ブロック）を配置可能
- 知識獲得後の ANN による関数近似が可能

本シミュレータには数値計算モードと，内部状態描画モードがあり，適宜選択してシミュレータを実行可能である．内部状態描画モードにおける，追跡問題の画面上のスクリーンショットを図 A.1 に示す．

本研究においては，上記シミュレータを Lenovo 社製 Think Server TS140 上で実行した．参考までに，Think Server TS140 の仕様を以下に記し，外観を図 A.2 に示す．

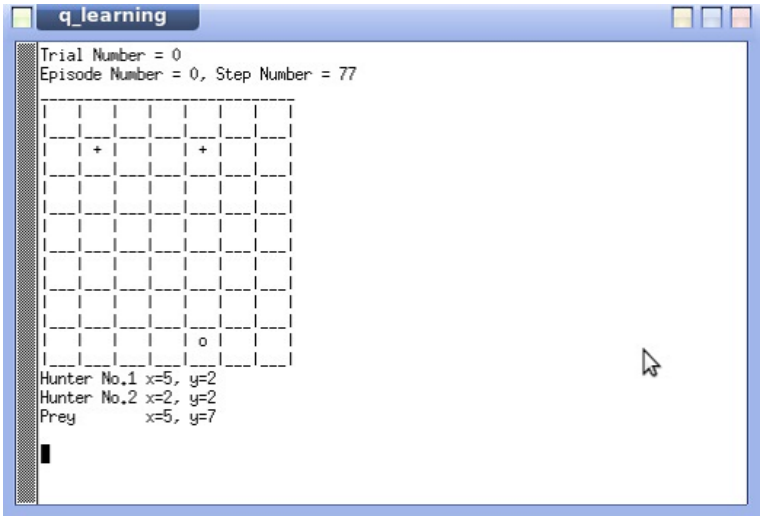


図 A.1 描画モードにおけるマルチエージェント強化学習シミュレータ

表 A.1 Think Server TS140 の仕様

Component	Name and value
Processor	Intel Xeon E3-1225 v3 prossesor (3.20GHz, 4 core)
Main memory	24 GBytes (8GB ECC DDR3-12800E × 3)
Disk drive	500 GBytes
Operating system	Scientific Linux 6.4 x86_64

図 A.2 Think Server TS140 (引用 : http://shopap.lenovo.com/jp/server/thinkserver/ts-series/ts140/)

付 録B　Stineman関数による学習曲線の描画

本論文では，エージェントのパフォーマンスの評価に学習曲線や転移曲線，収束値曲線を用いている．転移曲線や収束値曲線は，学習曲線のデータの分析後にグラフを作成しているためデータ点数が比較的少ない．しかし，学習曲線は最大 10000 episode ものステップ数をグラフで描画するためデータ点数が多い．さらに，強化学習の確率的な挙動（学習が偶然成功するケースや偶然学習が遅いケース）を抑えるために 10 trial の試行を行い，それらのアンサンブル平均を学習曲線として描画している．

動的環境における学習曲線は，アンサンブル平均を行ってもステップ数の増減が激しく，いくつかの学習曲線を同時にグラフに描画すると曲線の特徴や傾向が把握しにくい（図 B.1）．そこで，本論文における一部の学習曲線は，スムージング処理の 1 つである Stineman 関数を適用して描画している [103]．これは，太く見える学習曲線の傾向を表示するためと，学習曲線が覆いかぶさることを回避するためである．本処理は学習曲線を近似して関数を明らかにすることを目的にしているわけではなく，あくまでも学習曲線の傾向を描画するために用いている．図 B.1 を Stineman 関数によりスムージング処理したグラフの例を図 B.2 に示す．図 B.2 は，本来の学習曲線のプロットと Stineman 関数によりスムージングを行った波形を示している．赤線（実線）と青線（破線）で互いの波形が重なることなく，学習曲線の傾向を見ることができる．

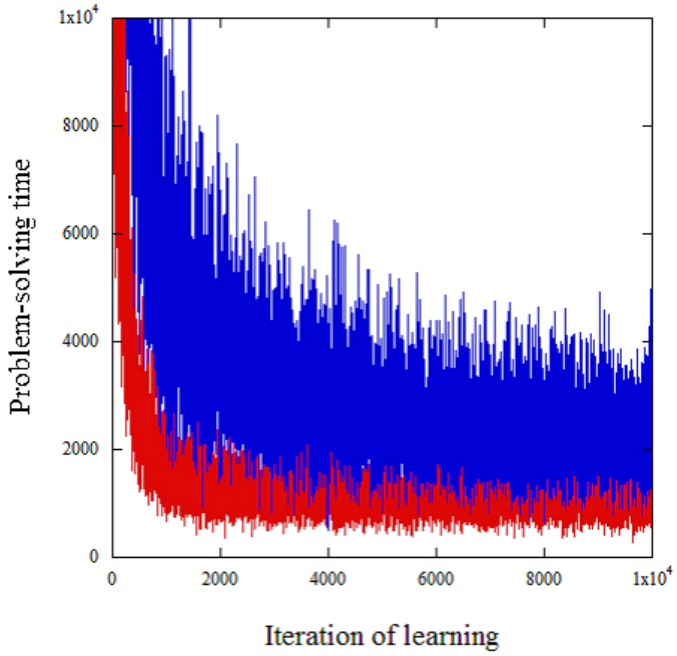


図 B.1 描画が重なる 2 つの学習曲線

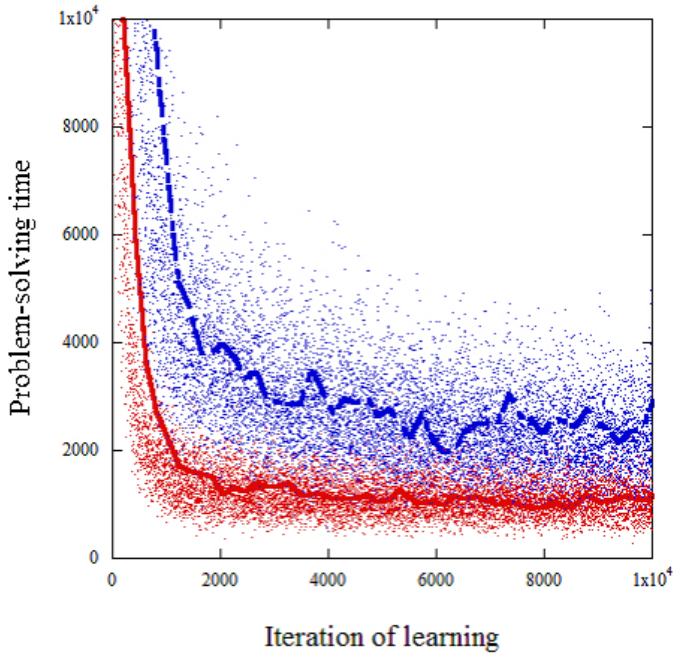


図 B.2 Stineman 関数によるスムージングの例

付 録C　小形汎用自律全方向移動ロボットプラットフォーム

C.1　概要

本ロボットは，研究室内実験を考慮して開発された小形汎用自律全方向移動ロボットプラットフォーム ZEN-Q である（図 C.1）．ZEN-Q は 4 輪のオムニホイールで構成された全方向移動台車を有し，各オムニホイールをそれぞれに接続されたモータにより駆動する．また，IA32 の汎用計算機を搭載し組込み計算機特有のクロス開発を用いないで，パーソナルコンピュータなどで動作プログラムを開発し，コンパイルした実行ファイルを直接ロボット内部で実行可能である．さらに，赤外線距離センサを 8 個，ロボットの周囲方向を測定する方向に実装されており，周囲の物体との衝突回避が可能である．通信機能として，無線 LAN への接続が可能であり，一般的に販売されているアクセスポイントを用いてロボットの通信環境の構築が可能である．

C.2　駆動系

ZEN-Q の駆動系は表 C.1 に示す部品から構成されている．駆動系のベースとして 4 つのオムニホイールと，それを駆動する 4 つのモータから構成されている．モータドライバは，産業技術総合研究所で開発されたモータコントローラと一体型の DC サーボモーターユニット A-10 の制御基板（サーボ基板）を実装している．サーボ基板は 1 つのモータを制御可能であり，磁気エンコーダによる車輪回転数の制御系へのフィードバックが可能である．PD 制御が可能である．

駆動系の設計値として，本体重量とペイロードを含む 2kg のロボットを最大移動速度 0.5m/s としている．しかし，プログラミングライブラリの速度指令値上限やモータからオムニホイールへの駆動力伝達効率などにより実際は約 0.1m/s で走行させている．

オムニホイールには，ダイセン電子工業の TD-046-01 を用い（図 C.2(a)），モータには S.T.L Japan の小型高出力モータ誉 31 を用いている（図 C.2(b)）．オムニホイールは

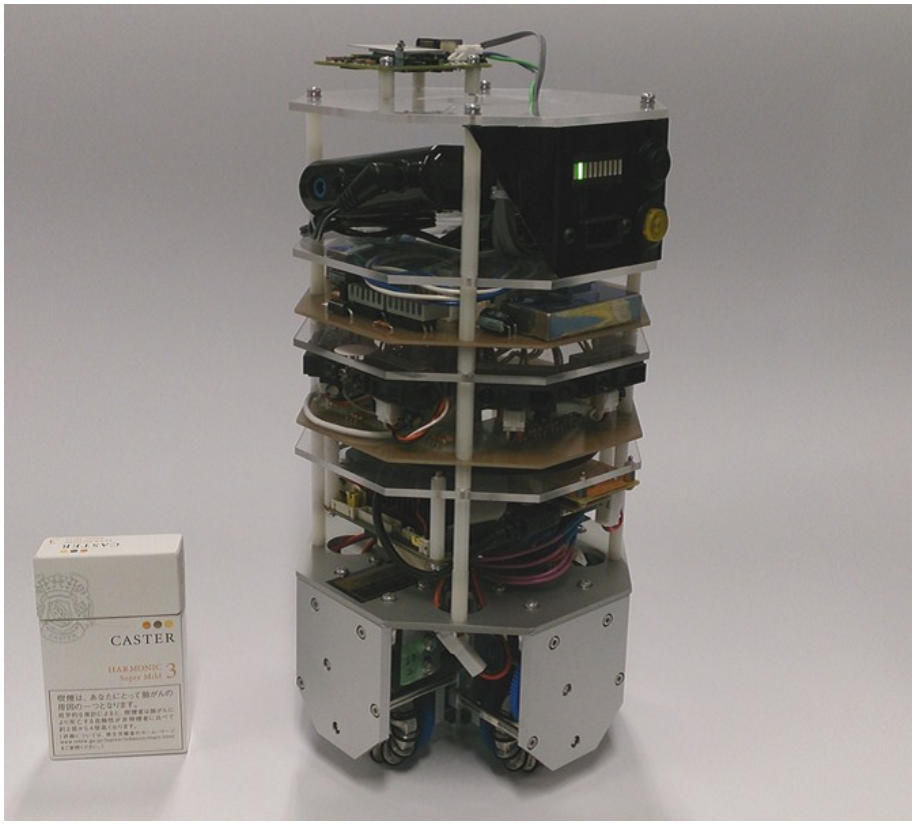


図 C.1 小形汎用自律全方向移動ロボットプラットフォーム ZEN-Q

MC ナイロン製のギアによりモータの駆動が伝達され，本ロボットでは減速比 30:1 として構成している．

C.3 電装系

ZEN-Q の電装系の仕様を表 C.2 に示す．電装系の各デバイスは図 C.3 に示すブロックダイアグラムのように，図 C.4 に示す電源制御ボードを中心に接続されている．

組込み計算機である PICO820 は，IA32 であるため他の計算機によりクロス開発を行う必要がない．開発用計算機の統合開発環境にてロボットのプログラムを開発し，実行ファイルをロボットに転送してプログラムの実行が可能である．

電源制御ボードは，バッテリの 12V 出力の電圧を駆動系と電装系の電源系統へ分配する機能が実装され，DC-DC コンバータにて電圧の降圧と安定化を行っている（図 C.4）．また，AC アダプタと内部バッテリを自動的に切り替える回路が実装されている．AC アダプタがコントロールパネル（図 C.5）を経由して電源制御ボードに接続されている場合，駆動系や電装系は AC アダプタから供給される電圧により駆動され，電圧供給を遮

表 C.1 ZEN-Q 駆動系の仕様

Component	Name and value
Type of wheel	TD-046-01(Omni-wheel)
Number of wheel	4
Motor	DC motor 誉 31
Number of motor	4
Gear reduction ratio	30:1
Motor driver and controler	DC サーボモーターユニット A-10 のサーボ基板
Number of A-10	4

(a) オムニホイール TD-046-01(引用 : http://www.daisendenshi.com/)

(d) モータ 誉 31 (引用 : http://stljapan.com/product03.html)

図 C.2 駆動系部品

断せずにバッテリ交換や長時間の作業を実現する．AC アダプタが切断された場合は，瞬停を発生させずにバッテリ駆動へと自動的に切り替わる．

図 C.5 に示す電装系用のコントロールパネルには，ZEN-Q 起動用のスイッチや内部に実装された計算機の起動スイッチ，AC アダプタ用ソケット，内部供給電源インディケータが実装されている．Main SW により ZEN-Q に実装された全てのデバイスが起動できる．Power SW of computer により計算機のみの起動や，再起動などが行える．A/C adapter socket に 12V 出力の AC アダプタを接続することにより，ZEN-Q の電源リソースが内部バッテリから外部電源へと自動的に切り替えられる．既に述べたが，瞬時に内部電源と外部電源を切替えることで，バッテリ交換の際に電源を切る必要が無く，また長時間のプログラミングや動作検証作業などの時にバッテリ残量に依存しない作業が行える．Power supply indicator (PSI) は ZEN-Q の内部に電圧がどこまで供給できているか示すステータ

表 C.2 ZEN-Q 電装系の仕様

Component	Name and value
Computer	PICO820 (x86 embedded computer)
Battery	Energizer XP8000A (8000mAh)
Communication device	FX-DS540-STB-ML (Ethernet converter)
Sensor	GP2Y0A21 (PSD sensor)
Servo I/O	DXHUB BTE068B

スバーである．図 C.5 の PSI において，未点灯の左から，“AC アダプタ”，“バッテリ”，“12V 電源供給”，“DC-DC コンバータ 5V 出力”，“DC-DC コンバータ 12V” 出力を意味している．各インディケータが点灯していれば電圧が正しく供給されていることを意味する．消灯している場合は，電源が接続されていないか故障を意味する．

ZEN-Q には周囲の距離を測定する赤外線 PSD センサが 8 機搭載されている．これらのセンサから取得された距離情報は，図 C.6 の内部状態表示機能付き I/O ボード（以下 I/O ボード）を介して ZEN-Q に内蔵されている計算機へ情報が送信される．I/O ボードは，8 角形の各辺に RGB の 3 色の LED が各色 2 つずつ実装され，ZEN-Q の内部状態を色により表示する機能を備えている．例えば，通常状態は緑の LED を発光させ，壁などの障害物を検知した場合は赤の LED を発光させるなどの運用が可能である．

本ロボットと他のデバイス（計算機や他ロボット）との通信には無線 LAN を用いる．組込み計算機の PICO820 には無線 LAN デバイスが実装されていないため，図 C.7 に示すイーサネットコンバータを実装した．これにより，市販の無線 LAN アクセスポイントを経由した通信が可能となる．

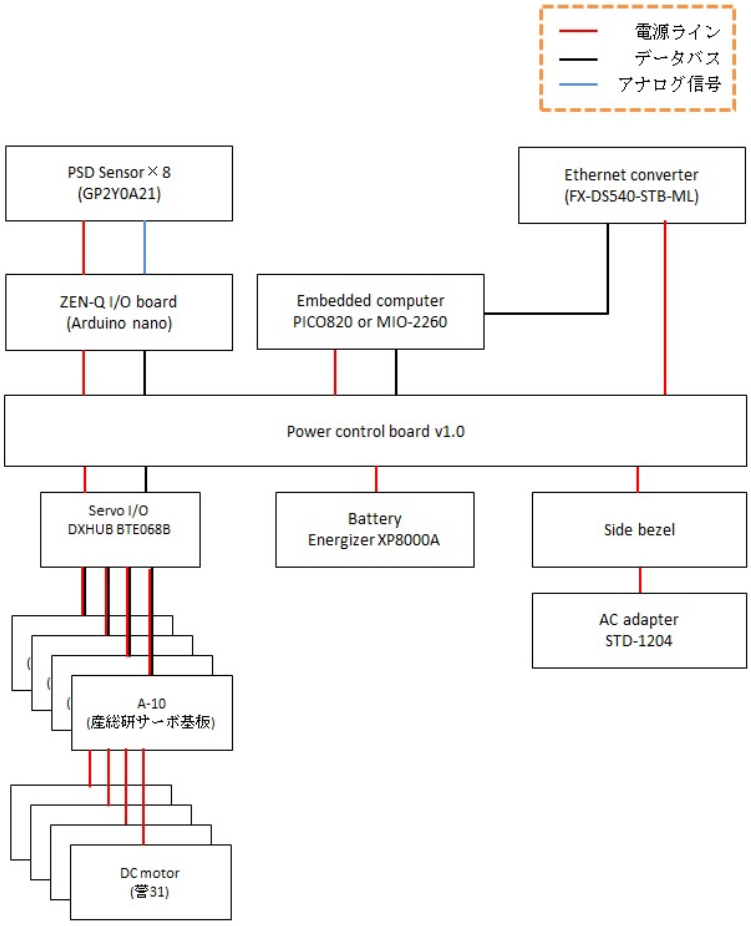
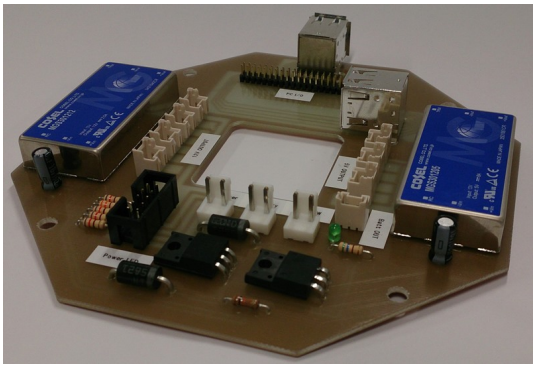
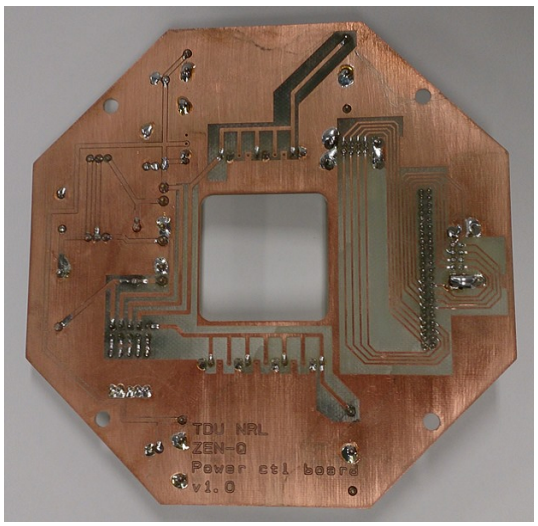


図 C.3 ZEN-Q の内部電装系ブロックダイアグラム



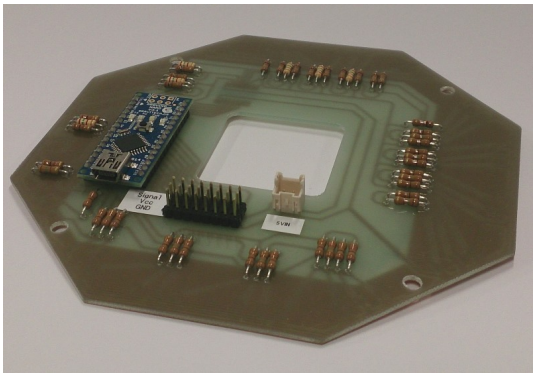
(a) 概観



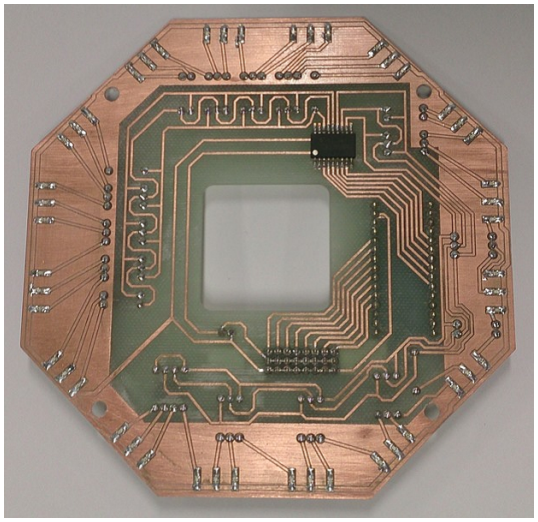
(d) PCB パターン

図 C.4 電源制御ボード

図 C.5 電装系用コントロールパネル



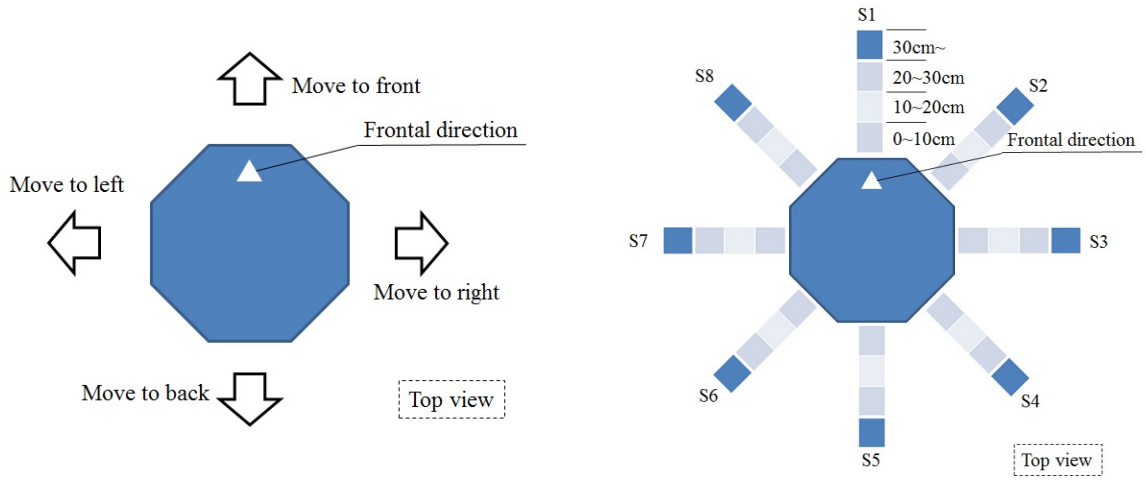
(a) 概観



(d) PCB パターン

図 C.6 内部状態表示機能付き I/O ボード

図 C.7 イーサネットコンバータ



(a) 移動可能な方向　　(b) 取得可能なセンサ情報と距離

図 C.8 ロボットの行動と観測可能な環境状態

C.4　実ロボットによる CEFDn を用いた収束推定予備実験

本節では，前述の小形汎用自律全方向移動ロボットプラットフォーム ZEN-Q を用いて，第 5 章に示した収束推定手法の有用性を評価する．第 5 章では，計算機実験による評価のみであり，実ロボットを用いた場合の有用性評価が行えていない．本節では，実ロボットによる CEFDn を用いた収束推定の可能性を評価する．評価関数としては，最短経路問題を採用する．

C.4.1　使用ロボット

本実験では，前述の自律全方向移動ロボットプラットフォームを用いる．ロボットは，図 C.8(a) のように基準方向に対し前後左右に移動できる．また，図 C.8(b) のようにロボットの周囲 8 方向に赤外線距離センサが搭載され，距離に応じた 4 分解能で周囲の障害物を認識可能である．

ロボットは，0.5 秒間隔で行動を選択し，秒速約 80mm/sec で移動することが可能である．強化学習として Q 学習のアルゴリズムが実装されており，学習した方策はロボット内のメインメモリ内に保存する．行動選択関数としては，ボルツマン分布モデルを実装している．強化学習に用いるパラメータは，表 C.3 に示す値をロボットに設定している．

表 C.3 学習パラメータ

Parameter	Value
Learning rate α	0.1
Discount rate γ	0.99
Reward r	1
Boltzmann parameter T	0.005
Default Q-value	0

表 C.4 CEFDn のパラメータ

Parameter	Value
Box sizes δ	1, 5, 10, 50, 100
Sweep interval p	1 episode
Analysis windows size e (horizontal)	100 episode
Threshold value D_{TH}	1.5
Sequential updating number N	300

C.4.2 実験環境と条件

本実験の最短経路問題は図 C.9 のように設置した．ロボットのスタート地点やゴール地点，フィールド内の大きさは図 C.10 の通りである．本実験の環境は，静的環境ではなくロボットの移動誤差やセンサ誤差，照明条件等を考慮すると動的環境であると言える．

ロボットは，必ずスタート地点から移動を開始し学習を行う．ゴール地点に到達すると，報酬が獲得できるようにプログラムされている．ゴール地点に到達したロボットは，手動でスタート地点に戻され，次エピソードを実行する．上述の一連の流れを 3000 episode 行い，CEFDn により収束推定した時点でのステップ数と，全てのエピソードを終了し，十分に学習した状態のステップ数を比較する．これにより，収束指定した時のパフォーマンスと，その後学習を継続しても強化学習によるパフォーマンスの改善が行われない，すなわち学習曲線の収束を確認する．本実験では，上述の実験手順を 1 回行う．

本実験における，CEFDn のためのパラメータ設定は表 C.4 に示す．これらの値は，本論分の計算機実験から有用性が示されたパラメータである．

C.4.3 実験結果・考察

ロボットが強化学習することで出力される学習曲線を図 C.11 に示す．図 C.11 では，学習と同時に獲得される学習曲線のデータから算出した，CEFDn によるフラクタル次元も示している．

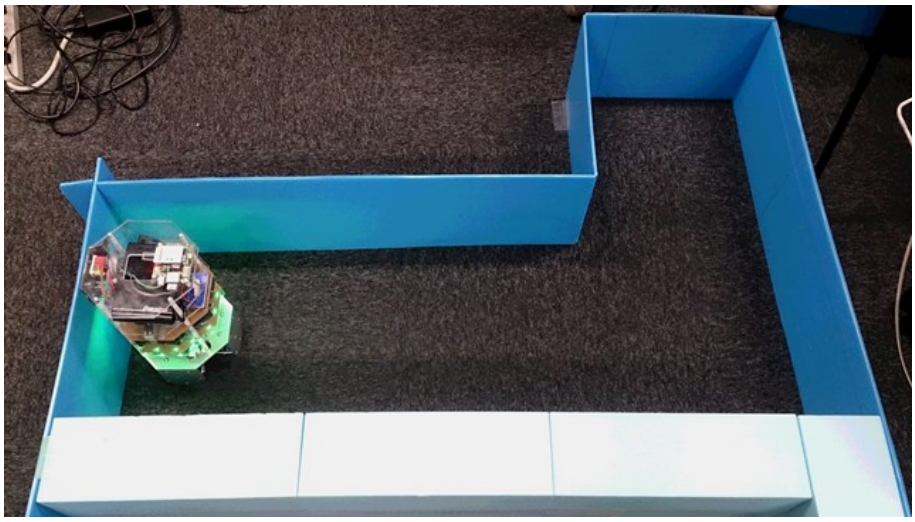


図 C.9 実環境における最短経路問題

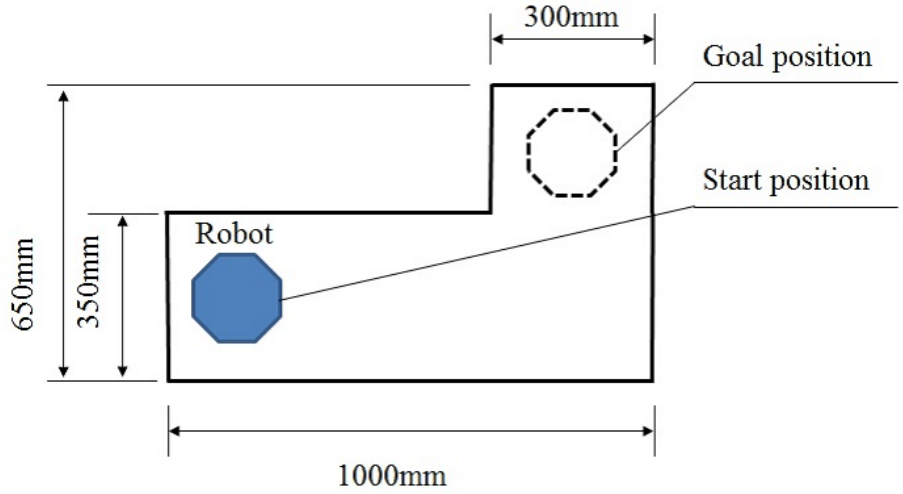


図 C.10 実験環境の寸法 (top view)

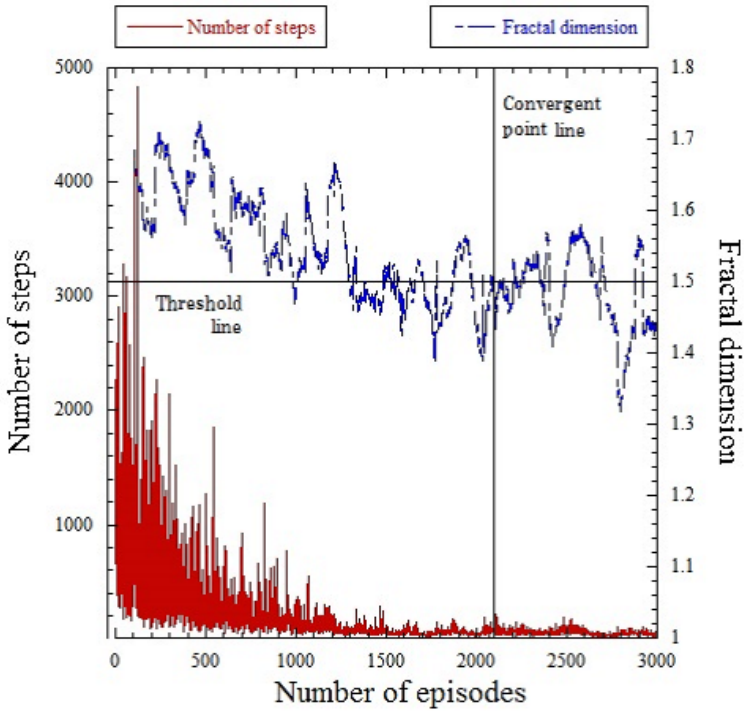


図 C.11 実ロボット実験における学習曲線とフラクタル次元の推移

図 C.11 から，学習曲線は振動しながら収束の傾向が現れており，最短経路問題の最短経路を強化学習により獲得したと見て取れる．学習初期における，最も長いゴールまでの所要時間は約 40 分，収束時におけるゴールまでの所要時間は約 15 秒であった．また，フラクタル次元の推移は上下の振動とともに，降下傾向が発現し閾値である D_{TH} を下回った．

CEFDn により収束を推定したエピソードは 2101 episode であり，その時のステップ数は 72.01 step（直前 100 episode 平均）であった．また，3000 episode の強化学習を実行し，目視による収束確認時のステップ数は 50.84 step であった．それらの誤差率は，29.40%であることから約 30%の誤差はあるが，学習初期のステップ数から比較すると十分にパフォーマンスが高くなってから収束を推定し，CEFDn による自律的な収束推定が可能であることが実験により示唆された．

付 録D　クラウド模擬システムの構成

本付録では，第 6 章で用いたクラウド模擬システムや，そのシステムに実装されている分散処理フレームワークの Hadoop に関して概説する．

本論文で構築したクラウド模擬システムは，一般的にクラウドコンピューティングで用いられているシステムを再現し，強化学習エージェントが獲得した知識を外部計算機に蓄積し，統合することを目的に構築した計算機ネットワークシステムである．

D.1　Apache Hadoop とネットワークシステム

クラウドコンピューティングとしての機能を実装するために，Apache Hadoop (以下 Hadoop) を用いたマルチノードクラスタを構築した．Hadoop のマルチノードクラスタは計算処理の分散を可能とし，計算機の増設によるリソースのスケールアウトが容易に行えることが利点としてあげられる．Hadoop は米 Google 社の AWS などのサービスでも提供され，クラウドコンピューティングとして標準的なフレームワークである．Hadoop を実装したネットワークシステムは，Master server と呼ばれる Name server と，Slave node と呼ばれる Data node から構成される．本論文で使用したクラウド模擬システムでは，図 D.1 のように Name node を 1 台，Data node を 2 台使用した．Hadoop には Hadoop-2.4.0 のバージョンを使用し，Master node と Slave node 全て共通のバージョンである．

本システムでは，Name node を Lenovo 社製 Think Station W540 で実行し，Data node を HP 社製 ProLiant Micro Server Turion II NEO N54L で実行した．参考までに，Think Station W540 の仕様を表 D.1 に示し，外観を図 D.2 に示す．また，ProLiant Micro Server Turion II NEO N54L の仕様を表 D.2，外観を図 D.3 に示す．

D.2　マルチノードクラスタによる分散処理

Hadoop によるマルチノードクラスタでは，MapReduce を用いた計算処理の分散が可能である．マルチノードクラスタは，大きく分けて Master server と Slave server から構成され，Master server が分散処理を行うデータと Slave server を管理し，Slave server は実際の

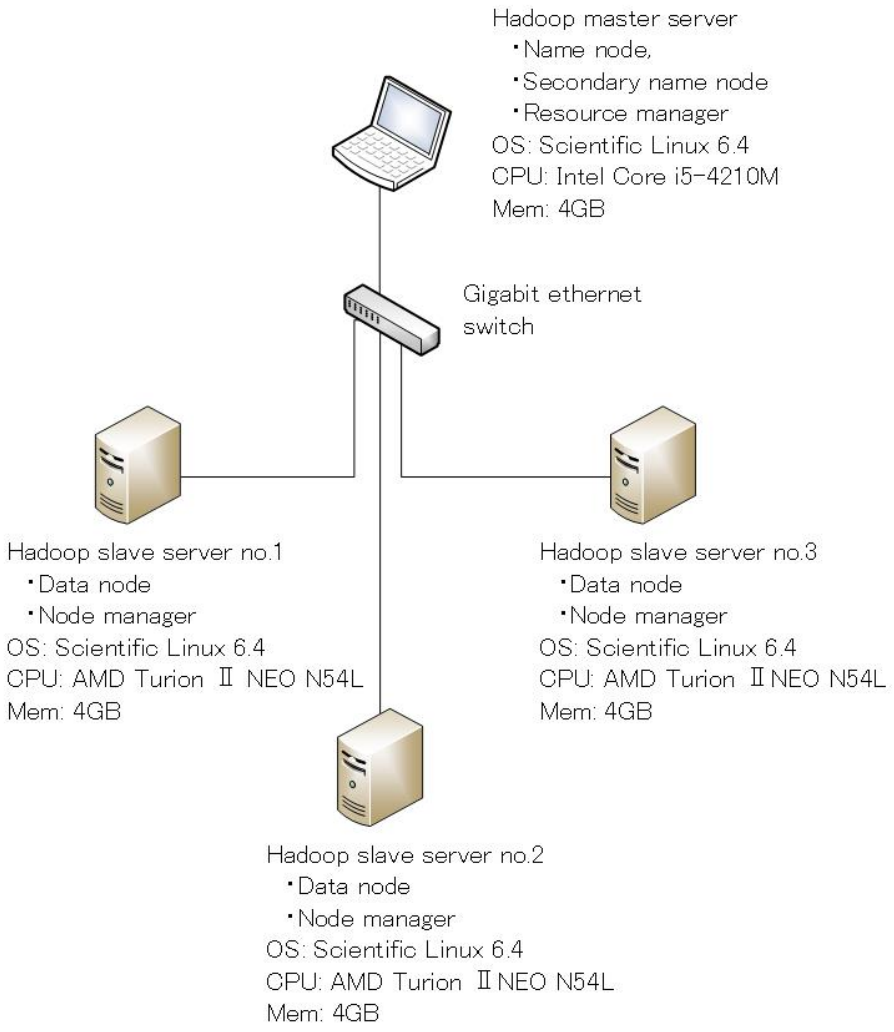


図 D.1 クラウド模擬システムのネットワーク構成図

表 D.1 Think Station W540 の仕様

Component	Name and value
Processor	Intel Core i5-4210M prossesor (2.60GHz, 4 core)
Main memory	4GBytes (PC3-12800 DDR3L × 3)
Disk drive	500 GBytes
Operating system	Scientific Linux 6.4 x86_64

図 D.2 Think Station W540 (引用 : `http://shopap.lenovo.com/jp/notebooks/thinkpad/w-series/w540/`)

データの処理を担当する [104]．特徴として，Slave server のマシン台数を多くすればマルチノードクラスタの処理性能は向上し，このスケールアウトもまた容易である．Hadoop は，従来の Relational database management system (RDBMS) と異なり，Hadoop file system (HDFS) という独自のファイルシステムを用いる．Hadoop で扱うデータは全て HDFS 上に保管され，また各 Slave server で確保されている HDFS 領域は，概念的には 1 つのディスク領域として処理される．すなわち，ユーザはマルチノードクラスタ上のどこにデータが保存されているかは意識しない．また，HDFS における Master server を Name node と呼び，Slave server を Data node と呼ぶ．

MapReduce は Map 処理と Reduce 処理により構成される．ユーザは Map 処理と Reduce 処理の具体的な処理内容を定義すると，MapReduce が自動的に組み込まれた処理を実行する．とあるファイル内の文字列における単語のカウント処理を例にすると Map 処理は「ファイル内の文字列の単語を抽出する」を実行し，Reduce 処理は「抽出された単語の同じものをカウントする」という処理を行う．これにより，あるファイル内の文字列を単語別に出現回数を明らかにすることが出来る．

表 D.2 ProLiant Micro Server Turion II NEO N54L の仕様

Component	Name and value
Processor	AMD Turion II NEO N54L prossesor (2.2GHz, 2 core)
Main memory	4GBytes (PC3-10600E DDR3 × 3)
Disk drive	500 GBytes (SATA)
Operating system	Scientific Linux 6.4 x86_64

図 D.3 ProLiant Micro Server Turion 2 NEO N54L (引用 : http://h50146.www5.hp.com/products/servers/proliant/micro/)

D.3 マルチノードクラタを用いた知識統合予備実験

前節まで概説した Hadoop によるマルチノードクラスタを用いて，第 6 章で述べた知識統合の予備実験を行う．本実験の目的は，複数個のファイルを統合する処理に対して，マルチノードクラウスタのノード台数がどれくらい統合速度に対して効果的か評価することである．

D.3.1 実験条件

本実験では，D.1 節に示した Hadoop を用いたマルチノードクラスタを用いる．Slave server は図 D.1 に示した通り 3 台の Data node 用いる．統合する知識は，ファイルとして Master server 内に保存されており，本論文で得られた知識データを利用する．知識データは 1176490 行の Look-up table である．10 個，100 個，1000 個の知識データ統合処理をマルチノードクラスタで実行し，Data node を 1 台から 3 台のそれぞれの構成で統合速度を評価する．これにより，世界中の強化学習エージェントやロボットが共有し，それ

表 D.3 マルチノードクラスタによる知識統合速度

Number of knowledge	Times [sec] One data node	Two data node	Three data node
10	167	98	105
100	1404	801	605
1000	13579	8049	6281

らの知識データの統合処理を前提とし，クラウドコンピューティングである無数の Data node で統合処理することで，統合処理時間が問題とならないことを示す．

それぞれの知識データ個数に対する，マルチノードクラスタによる統合実験回数はそれぞれ 1 回おこなう．本実験で用いるプログラムは，Hadoop のライブラリに計算するファイルを指定するだけであり，一度に読み込むファイル数や順序，計算のための変数設定などは，Hadoop のライブラリが自動的に設定する．また，知識統合法は式 (6.4) と同等である．

D.3.2 実験結果

実験結果を図 D.4 と表 D.3 に示す．知識データの量が少ない場合，Data node の数に対して大きな差は現れていない．しかし，知識データ数が増加すると，Data node を増やすとともに計算時間も短くなることがみてとれる．具体的には，1 台の Data node d で要した計算時間を基にすると，2 台の Data node では，計算時間が約 42%減少し，3 台の Data node においては，約 55%の計算時間削減となった．それぞれのグラフは，両対数グラフにおいて良好な直線性を確認できる．

統合する知識データが少ない場合，Master server や Slave server の内部処理遅延，通信遅延，分散処理のシーケンスなどがボトルネックとなり，マルチノードクラスタの性能が大きく現れなかったと考えられる．また図 D.4 から，知識データ数に対して Data node の数は多い方が計算時間の優位性があると言える．1 台の Data node の計算時間に対して，単純に 2 台の Data node の計算時間が $\frac{1}{2}$ とならないが，両対数において，Data node の台数が多いほど，計算時間の直線が平行のまま低くなると見て取れる．Data node の台数に対して，処理時間の減少は比例関係にあると考えられる．したがって，本実験ではクラウドを模擬する小規模なマルチノードクラスタであったが，実際に運用されている大規模計算機リソースを用いれば，計算時間の減少に貢献できると言える．

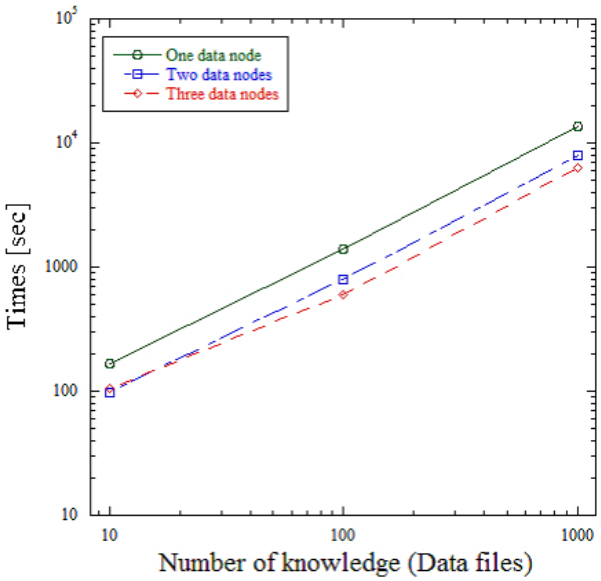


図 D.4 マルチノードクラスタによる知識統合速度比較