Vol.2015-ICS-179 No.16
2015/3/20

複数のタスク分布を考慮したマルチタスク強化学習問題への連想記憶モデルの応用

金原 史浩[1,a)]　加藤 昇平[1,b)]

概要：強化学習問題においてタスクを効率的に学習する手法の一つとしてマルチタスク強化学習がある．先行研究においてマルチタスク強化学習問題の定式化が提案されているが，タスク集団の分布が単一である場合しか考慮していない．本稿では，より一般的な問題を表せるように定式化されたマルチタスク強化学習問題を複数のタスク分布を考慮するように拡張する．更に，拡張されたマルチタスク強化学習に対して有効な手法として連想記憶モデルを応用したエージェントを提案し，その有効性をシミュレーション実験により確認する．

1. はじめに

強化学習は試行錯誤を通じて自律的に行動学習を行う機械学習の枠組みである [1]．強化学習には与えられたタスクの解決を行えるようになるまでに時間がかかるという問題がある [2]．強化学習の効率化のための手法の一つとして，マルチタスク強化学習（Multi-Task Reinforcement Learning 以下 MTRL）が挙げられる [3]．MTRL 問題ではエージェントが複数のタスクを学習することを考慮するため，エージェントが過去のタスクの経験を利用して新たなタスクを効率的に解くような学習を考えることができる．田中らによる先行研究 [3] では，マルチタスク強化学習（Multi-Task Reinforcement Learning 以下 MTRL）の研究が少ない理由に具体的な問題の定式化が為されていないことを挙げ，MTRL 問題の定式化を行い，更にその上で MTRL 問題に有効な手法を提案している．この定式化された MTRL 問題において，タスクは BV-MDPs（MDPs with Biased-Values）と呼ばれるタスクの分布から確率的にサンプリングされる．BV-MDPs は不変であり，単一の分布からタスクはサンプリングされ続ける．しかし実世界への応用を目標とするとき，常に単一の分布からタスクがサンプリングされるという仮定は不十分と考える．そこで，MTRL 問題を複数の BV-MDPs を考慮できる形式に拡張する．更に，拡張 MTRL 問題に有効な手法として連想記憶モデルを応用したエージェントを提案する．

[1] 名古屋工業大学
Nagoya Institute of Technology, Dept. of Computer Science and Engineering, Graduate School of Engineering, Gokiso-cho, Showa-ku, Nagoya-si 466–8555, Japan
[a)] kinbara@katolab.nitech.ac.jp
[b)] shohey@katolab.nitech.ac.jp

2. 田中らによる MTRL 問題 [3]

文献 [3] において田中らは複数タスクを取り扱った強化学習の研究が少ない理由として具体的な問題の定式化が明確でないことを挙げ，MTRL 問題の定式化を行っている．MTRL 問題を定式化するにあたって，BV-MDPs と呼ばれるタスクの集団および，生涯獲得報酬の定義が行なわれている．

2.1 BV-MDPs

通常，強化学習問題においてエージェントが学習を行なうタスクには MDP（Markov Dicision Process）が用いられる．MDP においてエージェントは，時間ステップ毎に状態 s_t を観測し，行動 a_t を選択して実行し，次状態 s_{t+1} と報酬 r_{t+1} を受け取る．MDP は 4 組項 $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$ で定義される．$\mathcal{S}$ は状態の集合，$\mathcal{A}$ はエージェントの取りうる行動の集合，$\mathcal{P}$ は任意の状態と行動から次の遷移可能な状態への遷移確率の集合（状態遷移関数），$\mathcal{R}$ は任意の状態と行動から次の状態遷移したとき確率的に与えられる報酬の期待値の集合（報酬関数）．エージェントの目標は，各状態における期待獲得報酬を最大化するような方策 π を学習することである．期待獲得報酬は価値関数と呼ばれ，以下の式で定義される．

$$v^{\pi}(s) = E\{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdots | s_t = s, \pi\} \qquad (1)$$

$v^{\pi}(s)$ は状態 s において方策 π に従った時の期待獲得報酬であり，γ は割引率である．そして，最適価値関数（最大化された価値関数）v^* は以下のように定義される．

$$v^*(s) = \max_{\pi} v^{\pi}(s) \qquad (2)$$

MDP の 4 組項を定めると，全ての状態 s には最適価値関数

が一通りに定まることが知られている．そこで，タスクの集団の性質を，最適価値関数の分布（の集合）により記述する．このように設定されたタスクの分布である問題クラスを BV-MDPs と呼ぶ．各状態 s についての最適価値関数の分布は独立であると仮定する．BV-MDPs において最適価値関数の分布の分散が小さいような場合には，エージェントは過去の経験を活かして効率的な学習が可能であると考えられる．また，複数のタスクを考慮するため，タスクには継続時間 τ を定める．すなわち，BV-MDPs からサンプリングされるタスクは 5 組項 $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \tau >$ で定義される．

2.2 生涯獲得報酬

複数タスク設定下では，エージェントが得る全てのタスクを通じての総獲得報酬を考える必要がある．これを生涯獲得報酬と呼ぶ．エージェントが遭遇する全タスク数を N とすると，生涯獲得報酬 TR は以下のように表される．

$$TR = \sum_{i=1}^{N} \int_{t_{i-1}}^{t_{i-1}+\tau_i} r_t dt \tag{3}$$

$$t_i = \sum_{j=1}^{i} \tau_j \tag{4}$$

$$t_0 = 0 \tag{5}$$

N は総タスク数である．

2.3 MTRL 問題の定式化

以上で説明した BV-MDPs および生涯獲得報酬を用いて MTRL 問題を定式化する．MTRL 問題においてエージェントの目的は，タスク内での収益の最大化に加えて，生涯獲得報酬も最大化することである．エージェントは一生の間に，図 1 のように，N 個のタスクを τ 時間ステップ毎に逐次的に与えられる．ここで与えられる各タスクは，BV-MDPs から独立に確率的にサンプリングされたものである．

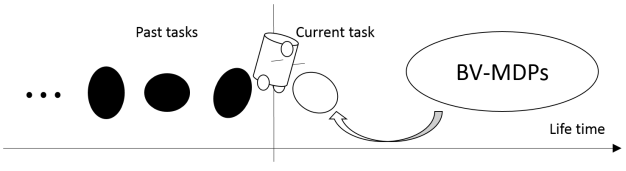

図 1　MTRL を行うエージェント [3]

3. MTRL 問題の拡張

前節で説明した MTRL 問題において，BV-MDPs は単一であり変化しない．しかし，実世界のモデル化としてこの定式化は限定的と考える．例えば，掃除ロボットが複合ビルを掃除する場合を考える．複合ビルは階によって，オフィスや店舗，宿泊施設などの異なる施設となっている．同一施設の階のみを考えるならば各部屋の内部構造は類似しているが，異なる施設の間での部屋の差異を考えると，類似していないことが多い．このような問題においてエージェントは，施設の違いを認識し，施設毎の経験情報を別々に保持した上で，経験情報を選択して活用することを考えなくてはならない．前述の複合ビルの問題での各施設は，それぞれ異なる BV-MDPs に対応すると考えられる．そこで，エージェントの一生の間において，タスクがサンプリングされる BV-MDPs が複数存在することを考慮するように MTRL 問題を拡張する．拡張 MTRL 問題においては図 2 に示すように，エージェントの一生の間に BV-MDPs が変化する．また，一度出現した BV-MDPs が再び出現することも考慮する．本研究においては，全 BV-MDPs は状態空間を共有すると仮定する．すなわち，BV-MDPs が変化したとしても状態空間は全てのタスクで共通である．また，MTRL 問題が有限時間を仮定しているため，エージェントが直面する BV-MDPs の数は有限であるとする．そして，エージェントは BV-MDPs が変化したことを明示的に与えられないものとする．

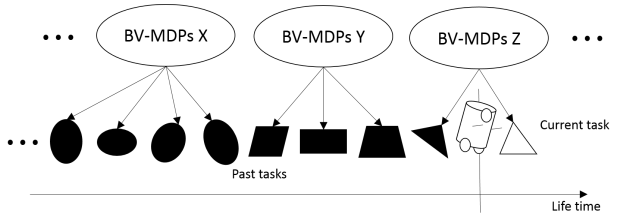

図 2　拡張された MTRL を行うエージェント

4. 提案エージェント

本章では，拡張 MTRL 問題に対して有効な手法として連想記憶モデルを応用したエージェントを提案する．MTRL 問題においては，エージェントは過去のタスクの学習経験を何らかの形で利用することで，生涯獲得報酬の最大化を目指すことが可能である．過去のタスク毎の経験を直接的に利用する場合，タスク数が増えるとメモリ量が爆発してしまうため得策ではない [3]．そこで，過去のタスクで得た経験を圧縮した情報を利用する必要がある．拡張 MTRL 問題においては，エージェントは経験を圧縮するだけでなく，BV-MDPs 毎に経験を保持・利用する能力が求められる．この問題に対して，連想記憶モデルが応用できると考える．連想記憶モデルは，Hopfield モデル [4] に代表されるニューラルネットワークの一種であり，ネットワーク内に二値のベクトル情報（パターン）を記憶すること（記銘）ができる．そして，パターンを入力として与えることで，入力パターンに近い記憶パターンを出力すること（想起）ができる．そこで，BV-MDPs 下での経験から得られる情報（以下 BV-MDPs 下の経験情報）と，その BV-MDPs からサンプリングされるタスクに関する情報を，従属的類似性があるようなパターンに変換する．その上で，BV-MDPs 下の経験情報を連想記憶モデルに記憶させれば，タスクに

関する情報を入力として，そのタスクがサンプリングされた BV-MDPs についての経験情報を連想記憶モデルから引き出すことができる．また，連想記憶モデルは類似したパターンを記憶させると，その混合状態が定常状態，すなわち想起可能な記憶となり，それが上位概念を表すことが知られている [5]．そのため，同じ BV-MDPs からサンプリングされたタスクの経験情報同士が類似パターンとなるようなパターンへ変換し，連想記憶モデルへと記憶させ続ければ，BV-MDPs についての経験情報を表すパターンの記憶が連想記憶モデル内に形成されると考えられる．

本来，BV-MDPs 下の経験情報は，タスクを与えられた際に引き出して利用しなければならない．しかしエージェントは，学習することでしかタスクの内容を把握できないため，タスクに関する情報とは，必然的にタスクについての経験情報になる．前述のように，BV-MDPs 下の経験情報を連想記憶モデルにより獲得した場合，当然，タスクの経験情報から BV-MDPs についての経験情報を引き出すことはできる．しかしタスクを効率的に解くための情報を，タスクを解かなければ得られないという問題が発生する．そこで提案エージェントは，現在のタスクと次のタスクが同じ BV-MDPs からサンプリングされるという仮定の下で経験情報の利用を行なう．すなわち，現在のタスクの経験情報から，そのタスクのサンプリングされた BV-MDPs 下の経験情報を引き出して次のタスクに利用する．

図 3 に提案エージェントの構造を示す．提案エージェントは，学習器，パターン生成器，メモリーの三つで構成される．学習器は強化学習によって，タスクから最適行動価値関数を獲得する部分である．パターン生成器は，タスクから得られた行動価値関数をパターンへと変換する．メモリーでは，タスクから得られたパターンを入力として，そのタスクがサンプリングされた BV-MDPs 下の経験情報を連想記憶モデルから引き出し，次のタスクへの利用のために学習器へと出力する．更に，タスクから得られたパターンを連想記憶モデルへ記憶することで，BV-MDPs についての経験情報を獲得する．また，タスクから得られる経験情報として，学習された行動価値関数を greedy な方策下で用いた場合の行動を用いる．価値関数の大きさに関する情報は失われるため，行動についての情報は完全にタスクの経験情報を表現しているわけではないが，タスク間で行動政策を再利用するような手法 [6] も存在するため，最終的に得られた greedy な行動をタスクの経験情報と見なすことは十分に有効であると考える．

4.1 学習器

学習器は強化学習によって，与えられたタスクを学習する．本稿では強化学習手法として，Q-learning[7] の派生である Dyna-Q[1] を用いる．また，推定対象である行動価値関数は，全状態-行動対の数と同じ大きさのテーブルに格納する．これを Q テーブルと呼ぶ．学習器は現在タスク終了後，タスクの経験情報としてパターン生成器へ Q テーブル

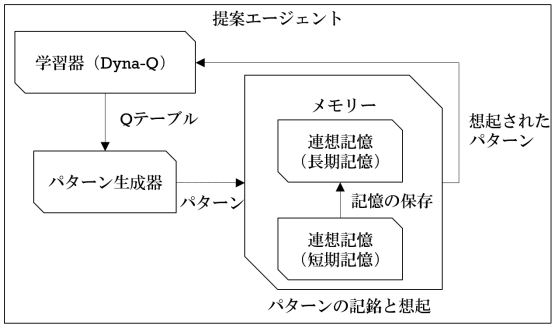

図 3 提案エージェントの構造

を出力する．その後学習器は次のタスクへ移行する前にメモリーからパターンを受け取る．そして受け取ったパターンを Q テーブルに変換し，次のタスクにおける Q テーブルの初期値としてセットする．但し，タスクの継続時間内に一度もゴールできなかった場合は，最適行動価値関数の推定が行われていないため，Q テーブルを出力せず即座に次のタスクへ移行する．パターンを受け取らなかった場合 Q テーブルは全て 0 で初期化される．パターンは 1 または -1 を要素とするベクトル情報である．エージェント内でやり取りされる全てのパターンの大きさは，全状態-行動対の数と同じである．パターンの要素のインデックスに基づいて，パターンの要素と状態-行動対は一対一対応している．パターンに基づいて Q テーブルは以下の式で初期化される．$\xi_{(s,a)}$ は受け取ったパターンの状態-行動対 (s,a) に対応する要素，$Q(s,a)$ は Q テーブルの状態-行動対 (s,a) についての値である．

$$Q(s,a) = \begin{cases} 1 & (\xi_{(s,a)} = 1) \\ 0 & (\xi_{(s,a)} = -1) \end{cases} \tag{6}$$

4.2 パターン生成器

パターン生成器はタスクが終了する度に，学習器から Q テーブルを受け取り，パターンへ変換する．ある状態 s について，取り得る状態-行動対 (s,a) の中で $Q(s,a)$ が最大となるような状態-行動対 (s,a) に対応する要素を 1 として，それ以外の場合を-1 としてパターンに変換する．パターン生成器は生成したパターンをメモリーへ出力する．

4.3 メモリー

連想記憶モデルにおいて，何度も記銘されたパターンは，想起されやすい記憶となり，記銘回数の少ないパターンの想起を阻害する．そのため，新しい BV-MDPs についての知識と，過去に遭遇した BV-MDPs についての知識を同一の連想記憶に保存するのは好ましくない．そこで，メモリーを図 3 で示したように，二つの連想記憶モデルから構成する．それぞれ「短期記憶」,「長期記憶」と呼ぶ．長期記憶には，パターン生成器から受け取ったパターンの記銘は行わず，短期記憶から記憶を移動させる．移動は，短期記憶の結合重みを長期記憶へそのまま追加することで行

う．記憶を移動する際の短期記憶の素子 i,j 間の結合重みを W_{ij}，μ 回目に記憶を移動する際の長期記憶の素子 i,j 間の結合重みを $J_{ij}(\mu)$ とすると，以下の式に従い長期記憶の結合重みを変化させる．

$$J_{ij}(\mu) = J_{ij}(\mu - 1)W_{ij} \tag{7}$$

長期記憶における想起は同期更新で行う．また，拡張 MTRL 問題において，エージェントは複数の BV-MDPs に直面するが，各 BV-MDPs の継続期間が同じとは限らない．長期間継続する BV-MDPs があった場合，その BV-MDPs 下の経験情報を記憶するための結合が強くなりすぎてしまい，長期記憶へ記憶を移動した場合，他の BV-MDPs の知識の想起を阻害することが考えられる．そこで，短期記憶には，ベータ次減衰するシナプスを持つ連想記憶モデル [8] を用いる．このモデルにおいて，減衰次数 β を正の値にし，減衰速度を α を調整することで結合が強くなりすぎるのを防ぐことができる．メモリーはパターンを受け取ると，パターンを短期記憶と長期記憶に入力して想起を行う．そして短期記憶と長期記憶から想起された各パターンと，入力パターンの間のオーバーラップをそれぞれ計算する．オーバーラップはパターン間の類似度の尺度であり，オーバーラップが 1 に近いほど，二つのパターンは類似している．N を連想記憶の素子数，ξ_i^1 および ξ_i^2 をそれぞれ，パターンの i 番目の要素とするとき，オーバーラップ m $(-1 \leq m \leq 1)$ は以下の式で定義される

$$m = \frac{1}{N}\sum_{i=1}^{N} \xi_i^1 \xi_i^2 \tag{8}$$

オーバーラップが閾値 θ 以上であるなら，現在のタスクがサンプリングされた BV-MDPs についての経験情報を想起できたと見なす．メモリーは，短期記憶から BV-MDPs についての経験情報を想起できなかった場合，BV-MDPs が変化したと判断する．この時，前回まで直面していた BV-MDPs が初めて直面した BV-MDPs であった場合，短期記憶の記憶を長期記憶へ保存する．BV-MDPs が変化した際，短期記憶はリセットされる．また，長期記憶から BV-MDPs についての経験情報を想起できなかった場合，そのタスクのサンプリング元の BV-MDPs が，初めて直面している BV-MDPs であると判断する．短期記憶または長期記憶から，BV-MDPs についての経験情報を想起できた場合，想起されたパターンを学習器へ出力する．両方から想起できた場合は，以前獲得したの経験情報の再利用を図った方が効率が良いと考えられるため，長期記憶から想起されたパターンを優先する．

5. 実験

提案エージェントが拡張 MTRL 問題に対しての有効性を検証するため，拡張 MTRL 問題の例を作成し実験を行う．提案エージェントの比較対象として，通常の Dyna-Q エージェント，文献 [3] で提案された手法の一つである Q テーブルのタスク間平均を初期値バイアスとして利用する Dyna-Q エージェント（以下 AB エージェント），現在のタスクから得られたパターンをそのまま次のタスクの初期値バイアスに用いる Dyna-Q エージェント（以下 LTPB エージェント）の 3 手法を用意する．文献 [3] では手法効果の評価には，真の状態行動価値関数と推定された状態行動価値関数との間の MSE（自乗誤差）を用いている，しかしながら本実験においては，「エージェントが BV-MDPs の変化に対応できているか」というエージェントの一生を通した評価に注目するため，生涯獲得報酬の最大化と個々のタスク内総獲得報酬の最大化という観点から，4 種類のエージェントについて，全タスクにおけるタスク内総獲得報酬と累積総獲得報酬を用いて評価する．

5.1 問題の設定

実験のベースとして迷路問題を用いる．拡張 MTRL 問題においてエージェントが直面する BV-MDPs として，二つの BV-MDPs を用意し，それぞれ BV-MDPs-A，BV-MDPs-B と呼ぶ．BV-MDPs-A に対応する迷路として図 4 に示す迷路 A を，BV-MDPs-B に対応する迷路として図 5 に示す迷路 B を用いる．エージェントは黒マスや欄外に移動することはできない．エージェントが観測する状態は左上から右下に向かって各マスに割り振られたラベルであるとする．エージェントは東西南北の 4 方向に移動できる．エージェントはスタート位置 S に割り振られた状態からタスクを開始し，ゴール位置 G に割り振られた状態に達すると，報酬 100 を与えられスタート位置 S に戻される．BV-MDPs の定義である，タスク間の確率的な最適価値関数の変化を設けるために，全てのマスは遷移の際に，確率的に遷移が成功するようなマスとする．遷移に失敗するとエージェントはそのマスに留まる．各マスの遷移成功確率は，タスクサンプリング時に，各マス毎に正規分布から独立にサンプリングされる．本実験において，正規分布は平均 0.5，分散 0.5 のものを用いた．遷移成功確率はタスクサンプリング時に各マス毎に独立にサンプリングされる．一試行の間にエージェントに与えられるタスク数は 800 タスクとする．タスクのサンプリングは BV-MDPs-A から開始し，100 タスク毎にタスクをサンプリングする BV-MDPs を切り替える．つまり，エージェントに与えられるタスクがサンプリングされる BV-MDPs は，A ⇒ B ⇒ A ⇒ B ⇒ A ⇒ B ⇒ A ⇒ B の順番で変化する．また，1 タスクの継続時間 τ は 10000 行動とする．

5.2 エージェントの設定

全エージェントの行動戦略は ϵ-greedy とする．ϵ-greedy は，行動決定時に確率 ϵ で行動をランダムに選択し，確率 $1-\epsilon$ で最大行動価値の行動を選択するという方法である．表 1 に，提案エージェントおよびその他のエージェントについて使用した各パラーメータをまとめる．学習器で用いるパラメータについては，Dyna-Q におけるステップパラ

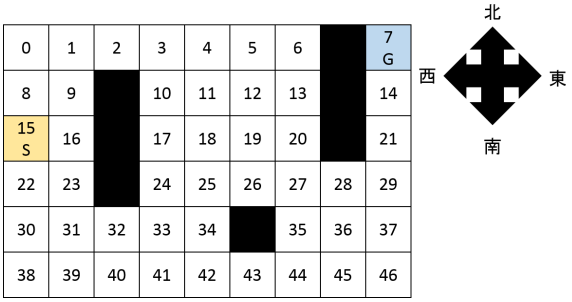

図 4 迷路 A：BV-MDPs-A に相当

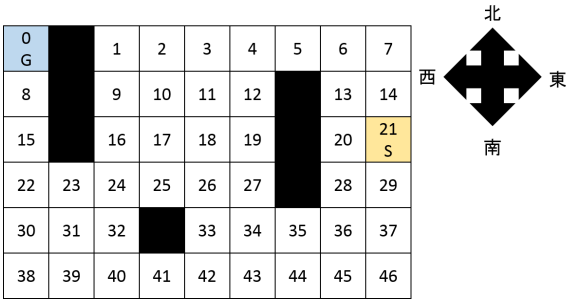

図 5 迷路 B：BV-MDPs-B に相当

メータを α_Q，割引率を γ，一回の行動毎に取る仮想行動の回数を X とする．メモリーで用いるパラメータについては，短期記憶と長期記憶の素子数を N，パターン間類似度の閾値を θ とする．また，短期記憶においてパターンを記銘する際に用いる減衰係数を α，減衰次数を β とする．

表 1 エージェントのパラメータ

パラメータの種類	提案	その他
確率 ϵ	0.5	
ステップパラメータ α_Q	0.1	
割引率 γ	0.95	
仮想行動回数 X	1	
素子数 N	188	-
オーバーラップの閾値 θ	0.35	-
減衰率 α	0.025	-
減衰次数 β	1	-

6. 実験結果

実験結果は全て 500 試行の平均である．実験において，タスクがサンプリングされる BV-MDPs は，100 タスク毎に切り替わるため，100 タスク毎に区切った区間にラベルをつける．BV-MDPs-A からタスクがサンプリングされる区間については順に A1～A4 とする．BV-MDPs-B からタスクがサンプリングされる区間については順に B1～B4 とする．図 6 および図 7 に，提案エージェントおよび AB エージェントについてのタスク内総獲得報酬をそれぞれ示す．また，図 8 は区間 A1 における全エージェントの平均タスク内総獲得報酬を一緒にプロットしたものである．全ての図において，横軸はタスク番号，縦軸はタスク内総獲得報酬である．また，表 2 に各区間内における各エージェントの平均累積総獲得報酬を，表 3 に各区間までの各エージェントの平均累積総獲得報酬をそれぞれ示す．

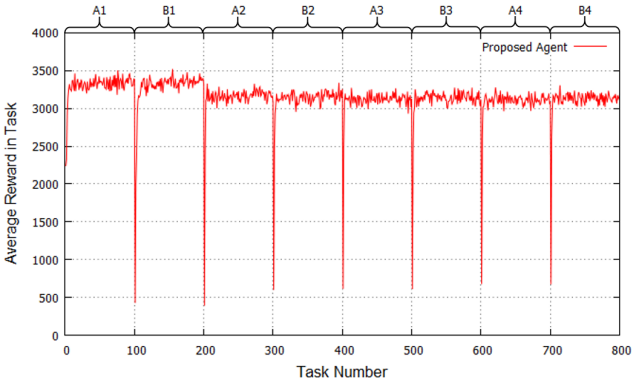

図 6 提案エージェントのタスク内総獲得報酬

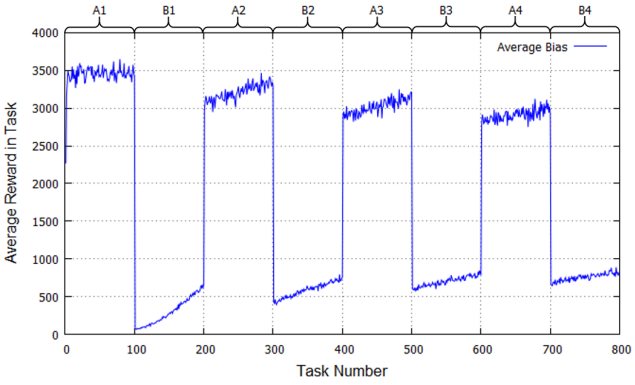

図 7 AB エージェントのタスク内総獲得報酬

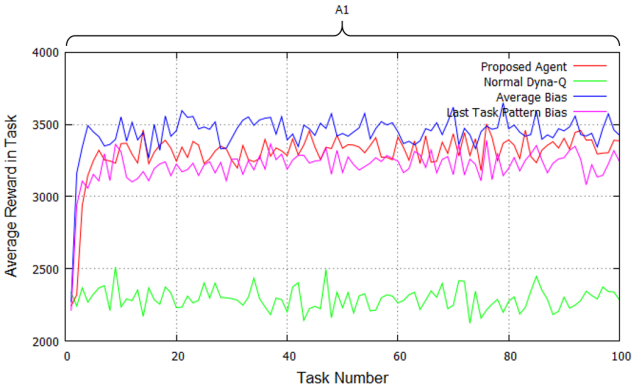

図 8 区間 A1 における全エージェントのタスク内総獲得報酬

区間 A1 の間は，通常の MTRL 問題と同等と見做せる．図 8 を見ると，タスク内総画得票集の値は提案エージェントはおよそ 3300，Dyna-Q はおよそ 2500，AB エージェントはおよそ 3400，LTPB エージェントはおよそ 3200 程度の値に収束していることが分かる．表 2 を見ても，提案エージェントの区間 A1 内での累積総獲得報酬は，Dyna-Q

表 2　区間内累積総獲得報酬

	提案	Dyna-Q	AB	LTPB
区間 A1	330258	229165.6	344399	321102.2
区間 B1	327630	229837.4	30206.2	316642.4
区間 A2	313144.6	228044.2	321003	319029.8
区間 B2	311528.8	228660.6	58789.4	319517.8
区間 A3	309326.4	228758.6	302020.8	319054.8
区間 B3	310768.6	228453	70211.8	317837.4
区間 A4	308396.2	228563.4	290837.4	319473.8
区間 B4	310663.8	228715	75783.6	317901.6

表 3　累積総獲得報酬

	提案	Dyna-Q	AB	LTPB
A1 まで	330258	229165.6	344399	321102.2
B1 まで	657888	459003	374605.2	637744.6
A2 まで	971032.6	687047.2	695608.2	956774.4
B2 まで	1282561.4	915707.8	754397.6	1276292.2
A3 まで	1591887.8	1144466.4	1056418.4	1595347
B3 まで	1902656.4	1372919.4	1126630.2	1913184.4
A4 まで	2211052.6	1601382.8	1417467.6	2232658.2
B4 まで	2521716.4	1830097.8	1493251.2	2550559.8

および LTPB エージェントを上回っている．これは，連想記憶モデルにより獲得される BV-MDPs についての経験情報の利用が有効に働いていることを示している．一方で，区間 A1 内で提案エージェントは AB エージェントほどのパフォーマンスは得られていない．これは，提案手法での BV-MDPs についての経験情報には，価値関数の大きさに関する情報が無いことが原因と考えられる．しかしながら区間 B1 を見ると，図 7 に示されるように，AB エージェントはパフォーマンスが大幅に低下していることが分かる．これは AB エージェントが BV-MDPs の変化に合わせた経験情報の利用ができていないためである．区間 A2 以降も，二つの BV-MDPs において平均を計算し続けることで，平均値利用の効果が失われていっており，AB エージェントはやがて通常の Dyna-Q と同程度のパフォーマンスに収束すると考えられる．一方，図 6 を見ると，提案エージェントは，BV-MDPs の変化時には誤った知識を利用してしまい大きくパフォーマンスが落ち込んでいるものの，BV-MDPs の変化に応じて利用する経験情報を変えることで，すぐにパフォーマンスを回復している．表 3 から分かるように，区間 B1 までの段階では，提案エージェントの累積総獲得報酬は AB エージェントの累積総獲得報酬を上回っている．このことから，提案エージェントの BV-MDPs の変化への頑健性が確認できる．しかしながら，提案エージェントのパフォーマンスは，区間 A2 以降若干低下している．区間 A2 以降で提案エージェントが経験情報として利用しているのは，長期記憶に保存されている区間 A1 で獲得された BV-MDPs-A についての経験情報，或いは区間 B2 で獲得された BV-MDPs-B についての経験情報である．BV-MDPs が切り替わった後のパフォーマンスの収束は，区間 A2 以降のほうが区間 A1，B1 よりも早いため，経験情報の再利用自体はできていると考えられる．にもかかわらず，区間 A2 以降で区間 A1 や区間 B1 ほどのパフォーマンスが発揮されないのは，長期記憶内で BV-MDPs-A と BV-MDPs-B についての二つの経験情報の記憶が互いに干渉し，崩れてしまっていることが原因と考えられる．区間 B2 以降において，BV-MDPs の変化直後のパフォーマンスの落ち込みが軽減されていることも，二つの知識が干渉し合っている影響と考えられる．

7. まとめ

本稿では，MTRL 問題の拡張を行い，拡張 MTRL 問題に有効な手法として連想記憶モデルを応用したエージェントを提案した．更に，提案手法の拡張 MTRL 問題への有効性をシミュレーション実験により検証した．その結果，連想記憶モデルにより獲得される経験情報の利用によるパフォーマンスの向上と，BV-MDPs の変化に対する頑健性を確認した．しかし，以前に獲得した経験情報の再利用がうまく行われていないことが分かった．今後，経験情報の再利用手法を検討する必要がある．また，タスクの BV-MDPs について学習する前に経験情報を引き出せるような手法を検討したい．

謝辞

本研究は，一部，文部科学省科学研究費補助金（課題番号 25280100，および，25540146），ならびに，JST 研究成果最適展開支援事業（A-STEP）ハイリスク挑戦プログラム（課題番号 AS2524004P）の助成により行われた．

参考文献

[1] Sutton, R. S. and Barto, A. G.: Reinforcement Learning, Bradford Book,(1998), 三上 貞芳, 皆川雅章訳，強化学習, 森北出版 (2000).
[2] Kaelbling, L. P., Littman, M. L. and Moore, A. W.: Reinforcement learning: A survey, *Journal of artificial intelligence research*, pp. 237–285 (1996).
[3] 田中文英，山村雅幸：MDP 集団の上におけるマルチタスク強化学習，電気学会論文誌 C (電子・情報・システム部門誌)， Vol. 123, No. 5, pp. 1004–1011 (2003).
[4] Hopfield, J. J.: Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the national academy of sciences*, Vol. 79, No. 8, pp. 2554–2558 (1982).
[5] Amari, S.-I.: Neural theory of association and concept-formation, *Biological cybernetics*, Vol. 26, No. 3, pp. 175–185 (1977).
[6] Fernández, F. and Veloso, M.: Probabilistic policy reuse in a reinforcement learning agent, *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, ACM, pp. 720–727 (2006).
[7] Watkins, C. J. and Dayan, P.: Q-learning, *Machine learning*, Vol. 8, No. 3-4, pp. 279–292 (1992).
[8] 宮田龍太， 青西亨， 綴木馴，倉田耕治：ベータ次減衰するシナプスをもつ連想記憶モデルの記憶特性，情報処理学会研究報告. MPS, 数理モデル化と問題解決研究報告, Vol. 2013, No. 14, pp. 1–6 (2013).