F-042

複数の目標を持つタスクに対する適格度トレースを用いた強化学習
Reinforcement Learning for Multiple-Goal Task with Eligibility Trace

小野寺 道寛 † 鈴木 輝彦 ‡ 太原 育夫 ‡
Michihiro Onodera Teruhiko Suzuki Ikuo Tahara

1. はじめに

強化学習とは，試行錯誤を通じて環境に適応する学習制御の枠組である [1]．強化学習において，エージェントは目標に到達する毎に報酬を与えられる．そして，エージェントは累積獲得報酬 (収益と呼ぶ) を最大にする行動を学習するので，同時に目標への到達方法を自動的に学習することができる．従来の強化学習の多くは目標が一つしかないタスクを対象に学習しているが，現実世界のタスクには複数の目標を持つものが少なくない．目標が複数あるタスクの場合，通常の強化学習アルゴリズムでは所持する情報が多すぎ，学習が収束するまで非常に時間がかかってしまったり，エージェントのメモリが足りなくなって学習自体が不可能になってしまうことがある (状態空間の爆発問題 [7])．

こうした問題を解決するためのアルゴリズムとして，GM-Q[2] や GM-Sarsa(0)[3] が既に提案されている．これらのアルゴリズムは，状態空間の爆発問題を解決することやそのタスクにおける出来る限り最大の収益を得ることを目的としており，複数の目標を持つタスクにおける学習の高速化自体を目的とした研究は殆んど行われていない．

そこで，本論文では適格度トレース [5] と呼ばれる全ての経験した状態行動対の Q 値を毎回更新する手法を利用することによって，複数の目標を持つタスクに対する学習の高速化を検討する．

2. 複数の目標を持つタスクに対する強化学習

2.1 GM-Q と GM-Sarsa(0)

単一目標のみのタスクの状態数と比較すると複数の目標を持つタスクの状態数は非常に多いため，状態空間の爆発問題が発生する．状態空間の爆発問題を解決するアルゴリズムとして GM-Q(GreatestMass-Qlearning)[2] がある．GM-Q では，まずタスクをエージェントと単一目標しか存在しないタスク (単純タスク) に分割するので，状態空間の次元数は目標の数に関係なく，状態空間の爆発を防ぐことが可能になる．目標が n 個存在するタスクの場合，n 個の単純タスク $T_i(i = 1, 2, \cdots, n)$ に分割し，各タスクに対して 1 個ずつ学習器を割り当てる．各学習器は Q 学習を行う．タスク T_i の時刻 t における状態を $s_{i,t}$，タスク T_i における Q 値を Q_i とする．タスク T_i の学習器は，状態 $s_{i,t}$ で行動 a_t を選択して状態 $s_{i,t+1}$ に遷移したとき，選択した状態行動対 $[s_{i,t},a_t]$ の Q 値 $Q_i(s_{i,t},a_t)$ を式 (1) のように更新する．

$$Q_i(s_{i,t},a_t) \leftarrow Q_i(s_{i,t},a_t) + \alpha[r_i + \gamma \max_a Q_i(s_{i,t+1},a) - Q_i(s_{i,t},a_t)] \quad (1)$$

ここで，r_i はタスク T_i において得られた報酬，$\alpha(0 < \alpha \leq 1)$ は学習率，$\gamma(0 \leq \gamma \leq 1)$ は割引率を表している．次に行動選択法だが，行動空間は分割せずに各タスクで共有するので各タスクは同じ行動を選択することになる．具体的な行動選択法は次のようになる．$1-\epsilon$ の確率で，式 (2) に従って行動を選択する．

$$\arg\max_a \sum_{i=1}^{n} Q_i(s_{i,t},a) \quad (2)$$

つまり，各タスクにおける Q 値の合計値が最も大きくなる行動が各タスクにおいて選択される．そして，ϵ の確率でランダム行動が行われる．このアルゴリズムを用いることによって，ある程度の収益を集めることが可能である．

このように GM-Q では，各タスク T_i の Q 値の更新の際に $\max_a Q_i(s_{i,t+1},a)$ すなわち Q 値の一番高い状態行動対を用いているが，各タスクにおいて Q 値の更新の際に用いられる状態行動対の行動 a は全タスクで同一とは限らない，むしろ同一でない場合の方が多い．つまり，使われることのない状態行動対を用いて Q 値を更新している可能性があり，Q 値は正しい期待報酬値を近似できていない．そのため，GM-Q では分割する前のタスクにおける最大収益あるいはそれに近い収益を得ることはできない．そこで，Sprague らは GM-Q の欠点を改善する手法として GM-Sarsa(0)(GreatestMass-Sarsa(0))[3] を提案した．この GM-Sarsa(0) は GM-Q をベースにしたアルゴリズムで，各タスクの Q 値の更新以外は全て同じである．このアルゴリズムにおいて，各学習器は Sarsa(0) を用いて学習する．つまりタスク T_i の学習器は，状態 $s_{i,t}$ で行動 a_t を選択して状態 $s_{i,t+1}$ に遷移したとき，次に行う行動 a_{t+1} を決めてから，選択した状態行動対 $[s_{i,t},a_t]$ の Q 値 $Q_i(s_{i,t},a_t)$ を式 (3) のように更新する．

$$Q_i(s_{i,t},a_t \leftarrow Q_i(s_{i,t},a_t) + \alpha[r_i + \gamma Q_i(s_{i,t+1},a_{t+1}) - Q_i(s_{i,t},a_t)] \quad (3)$$

この更新法を用いれば，次に必ず使われる状態行動対を用いて Q 値を更新できるため，Q 値は正しい期待報酬値を近似でき，多くの収益を集めることが可能となる．GM-Q よりも GM-Sarsa(0) の方が多くの収益を集められることは Sprague らの行った実験によっても証明されている [3]．

2.2 適格度トレースの適用

上記のアルゴリズムによって，複数の目標を持つタスクの学習の際に起こる状態空間の爆発を防ぐことができ，なおかつそのタスクにおける収益を多く集めることができる．本来，強化学習は試行錯誤を必要とするため学習速度が非常に遅いという欠点があり，これは実時間で処理を行わなければならない現実のタスクにおいても大きな障害となり得る．そのため，学習速度を上げることは非常に有用である．しかし，複数の目標を持つタスクを対象にしたアルゴリズムにおいて，純粋に学習速度を上げることを目的にしているものは少ない．ここでは学習速度の向上を図ることにする．学習速度の向上には，

†東京理科大学大学院理工学研究科情報科学専攻
‡東京理科大学理工学部情報科学科

- 学習の収束に要するエージェントの行動回数の減少
- 学習の収束に要する計算時間の短縮

の 2 種類が存在する [4]．ここでは GM-Sarsa(0) によって得られる収益と同程度の収益の獲得を前提とした高速化に着目し，GM-Sarsa(0) に適格度トレースを適用した GM-Sarsa(λ) を考える．

適格度トレース [5] とは，全ての経験した状態行動対の Q 値を毎回更新する手法であり，このアルゴリズムは，各タスクの Q 値の更新以外は全て GM-Sarsa(0) と同じである．Q 値の更新方法を以下に示す．時刻 $t = k$ において状態 $s_{i,k}$ で行動 a_k を選択して状態 $s_{i,k+1}$ に遷移したとき，全ての状態行動対 $[s_{i,x},a_y]$ の Q 値 $Q_i(s_{i,x}, a_y)$ を式 (4) のように更新する．

$$Q_i(s_{i,x}, a_y) \leftarrow Q_i(s_{i,x}, a_y) + \alpha\delta e(s_{i,x}, a_y) \quad (4)$$
$$(\delta = r_i + \gamma Q_i(s_{i,k+1}, a_{k+1}) - Q_i(s_{i,k}, a_k))$$

ここで，$e(s_{i,x}, a_y)$ を状態行動対 $[s_{i,x},a_y]$ に関するトレースと呼び，$e(s_{i,x}, a_y)$ の値は次の通りである．

$$e(s_{i,x}, a_y) \leftarrow \begin{cases} 1 & (s_{i,x} = s_{i,k}\text{かつ } a_y = a_k\text{の時}) \\ \lambda e(s_{i,x}, a_y) & (\text{それ以外の時}) \end{cases} \quad (5)$$

$e(s_{i,x}, a_y)$ の初期値は 0 なので，経験した状態行動対の価値しか更新されない．$\lambda(0 \leq \lambda \leq 1)$ を減衰係数と呼び，λ の値が大きければ，報酬から遠ざかっている状態行動対の価値に加算される値も高くなるので，速く学習が収束する．このように適格度トレースを用いることによって，用いない場合よりも少ない行動回数で，各状態行動対の Q 値は期待報酬値を近似することができる．しかし，このアルゴリズムをそのまま実装すると，毎回全ての状態行動対に対して Q 値の更新を行うので，学習の収束に要する計算時間自体は爆発的に増加することが予想される．そこで，適格度トレースを実装するには，計算時間を短縮するためのアルゴリズムも実装する必要がある．

2.3 計算時間短縮のためのアルゴリズム

ここでは，TD(λ) の対数時間更新算法 [6] を参考にして適格度トレースを使用した場合の Q 値更新時の計算時間を短縮するためのアルゴリズムを考える．このアルゴリズムでは，木構造 (二分木) を利用する．その木のノードには各状態行動対の Q 値に加算するための値を貯蓄しておき，葉ノードに 1 つの状態行動対の Q 値を格納する．状態行動対の Q 値の更新の時は，根ノードからその状態行動対の Q 値を持つ葉ノードまでの道のりで通過する全てのノードの貯蓄している値をその Q 値に加算する．つまりこのアルゴリズムでは，その時に使用した状態行動対の Q 値しか更新を行わないが，その時に加算される値は，その状態行動対が前回使用されてから今使用されるまでに適格度トレースで加算される値と同じ値になる．このアルゴリズムを用いることによって，多少各 Q 値の更新は遅れるものの，使用した状態行動対の Q 値にのみ適格度トレース使用時と同じ値が加算されるので，学習の収束に要するエージェントの行動回数の減少だけでなく学習の収束に要する計算時間の短縮も期待できる．このアルゴリズムの詳細は以下の通り

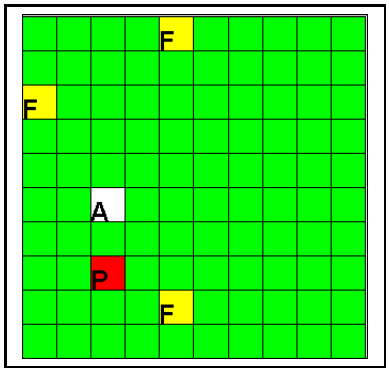

図 1: 餌集めタスク

である．

このタスクを単純タスク T_i とする．
ノード N は，最新減衰係数 d_N，後付け更新値 f_N という変数を所持する．また，葉ノード L は 2 つの変数だけなく，状態行動対 $[s_{i,x},a_y]$ の Q 値である $Q_i(s_{i,x}, a_y)$ も所持する．各変数の初期値は次の通りである．
$d_N \leftarrow 0$, $f_N \leftarrow 0$, $Q_i(s_{i,x}, a_y) \leftarrow 0$
状態行動対 $[s_{i,k},a_k]$ が使用されたとき，次のように $Q_i(s_{i,k}, a_k)$ が更新される．$Q_i(s_{i,k}, a_k)$ を所持する葉ノードを L_k とする．

1. $N \leftarrow \top$(N はノード，$\top$ は根ノードとする).
2. $f_N \leftarrow \lambda \times \alpha[r_i + \gamma Q_i(s_{i,k+1}, a_{k+1}) - Q_i(s_{i,k}, a_k)]$, $d_N \leftarrow \lambda$(λ は減衰係数)
3. 左右両方それぞれの子ノードの変数の値を更新する．$f_{N'} \leftarrow f_{N'} + d_{N'} \times f_N$, $d_{N'} \leftarrow d_N \times d_{N'}$ (N' は子ノードとする)
4. $f_N \leftarrow 0$, $d_N \leftarrow 1.0$, $N \leftarrow N'_k$ (N'_k は N' の内で $\top$ から L_k へのルートの通過点となる方のノードとする)
5. N が葉ノードでなければ，処理 2 に移る．葉ノードならば処理 6 に移る．
6. $Q_i(s_{i,k}, a_k) \leftarrow Q_i(s_{i,k}, a_k) + f_N + (1 - d_N)\alpha\delta$ $(\delta = r_i + \gamma Q_i(s_{i,k+1}, a_{k+1}) - Q_i(s_{i,k}, a_k))$
7. $f_N \leftarrow 0$, $d_N \leftarrow 1.0$

なお，この計算時間短縮のためのアルゴリズムを用いた GM-Sarsa(λ) を高速 GM-Sarsa(λ) と呼ぶことにする．

3. 実験

3.1 実験条件

3.1.1 餌集めタスク

高速 GM-Sarsa(λ) の有効性を検証するために，餌集めタスクを考える．餌集めタスクとは，図 1 のような 10×10 のグリッド空間において，エージェント (図 1 中の A) が捕食者 (図 1 中の P) から逃げながら，餌 (図 1

中の F) を拾っていくというタスクのことである．できるだけ多くの収益を集めることをこのタスクの目的とする．これは，報酬が本来の目的を達成出来るくらいにしっかりと設定されていれば，より多くの収益を集めることができる学習アルゴリズムの方が本来の目的をより高い水準で達成できるからである．また，強化学習におけるエージェントの本来の目的は最大収益を獲得すること [5] なので，この目的は妥当であると言える．

3.1.2 状態空間と行動空間

このタスクを単純タスクに分割せずに学習する時，エージェントはエージェント自身の位置，捕食者の位置，3 つの餌全てを考慮しないといけないので，エージェントの状態空間の次元数は 5 になり，その状態数は $(10 \times 10)^5$ と膨大な数になってしまって学習自体が不可能となる．そこで，次のようにタスクを分割する．

タスク 1 捕食者から逃げるタスク (エージェントが考慮すべき物 : エージェント自身の位置，捕食者の位置)

タスク 2 餌 1 を拾うタスク (エージェントが考慮すべき物 : エージェント自身の位置，餌 1 の位置)

タスク 3 餌 2 を拾うタスク (エージェントが考慮すべき物 : エージェント自身の位置，餌 2 の位置)

タスク 4 餌 3 を拾うタスク (エージェントが考慮すべき物 : エージェント自身の位置，餌 3 の位置)

このように分割することによって，状態数が $4 \times (10 \times 10)^2 = 4 \times 10^4$ となるので，学習することが可能になる．また，このときの行動空間 (エージェントが 1STEP 中に取り得る全行動パターン) は上下左右のいずれかへ 1 マス移動の 4 種類であり，各単純タスクはこの行動空間を共有する．つまり，各タスクにおいて同じ行動を行う．その他の実験条件は以下のとおりである．

- このタスクは非エピソードタスクとする．
- 餌を拾う (餌と同じマスに止まること) と，その餌は次の STEP には別の座標にランダムに配置される．
- 捕食者は常にエージェントに近づくように行動し，2STEP に 1 回行動する．
- 捕食者が 1 回の行動で行えるものは，上下左右のいずれかへ 1 マス移動とそのマスに停止の 5 種類．
- 捕食者に捕獲 (捕食者と同じマスに止まること) されても，学習は続くものとする．
- 餌を拾った時の報酬は 3，捕食者に捕獲された時の報酬は-5 とする．

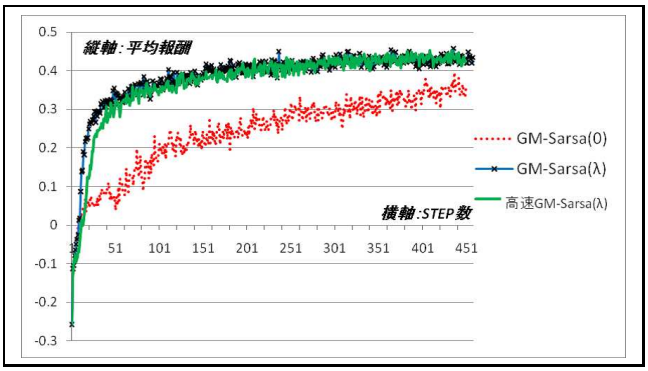

図 2: 各手法における STEP 数当りの平均報酬

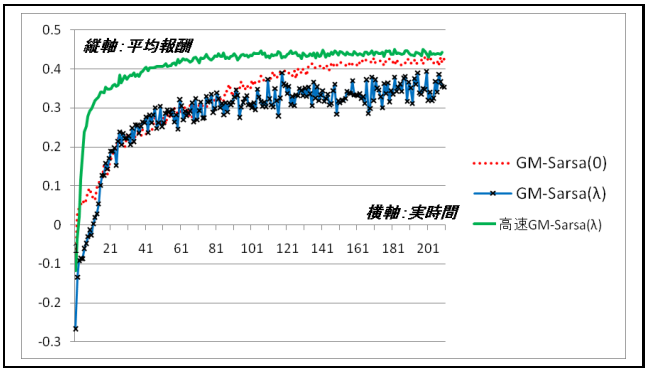

図 3: 各手法における実時間当りの平均報酬

3.2 実験結果

GM-Sarsa(0)，GM-Sarsa(λ)，高速 GM-Sarsa(λ) のそれぞれのアルゴリズムを用いて餌集めタスクの実験を実施した．その結果を，図 2，図 3 に示す．図 2 は STEP 数当りの平均報酬を表したグラフで，横軸は STEP 数 (単位 : 1 万 STEP)，縦軸は 1STEP で得られる平均報酬 (1 万 STEP の平均値) を表している．図 3 は実時間当りの平均報酬を表したグラフで，横軸は実時間 (単位 : 分)，縦軸は 1STEP で得られる平均報酬 (1 分間の平均値) を表している．図 2 の結果から，GM-Sarsa(λ) と高速 GM-Sarsa(λ) の方が GM-Sarsa(0) よりも少ない行動回数で学習が収束していることが分かる．また，計算時間短縮のためのアルゴリズムを用いたとき，そのアルゴリズムを使用しないときと比べて各 Q 値の更新が遅れるため，若干学習収束のために必要な行動回数が増えてしまうことも分かる．しかし，図 3 の結果から，GM-Sarsa(λ) は学習の収束に要する計算時間を増大させてしまうことが分かる．また高速 GM-Sarsa(λ) と GM-Sarsa(0) では，1STEP にかかる計算時間がほぼ同じなので，高速 GM-Sarsa(λ) の方が GM-Sarsa(0) よりも短い計算時間で学習を収束させることができる．以上より，高速 GM-Sarsa(λ) を用いることによって，少ない行動回数で且つ短い計算時間で学習を収束させることができると言える．

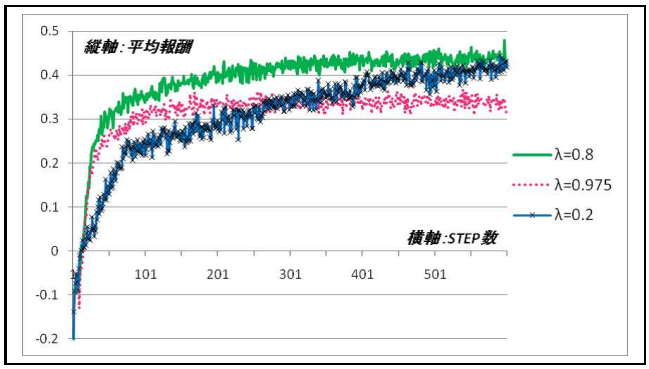

図 4: 餌を拾うタスクの λ の値を変えて実験

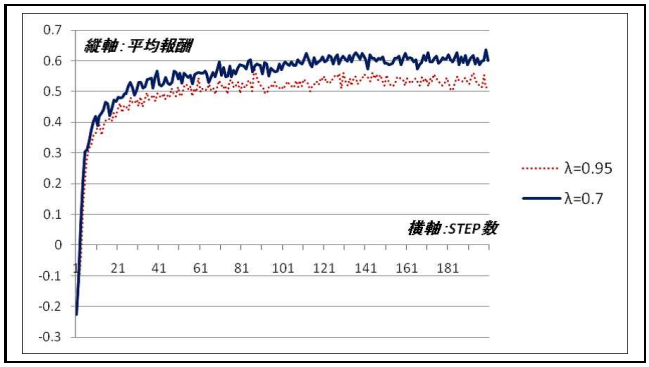

図 5: 7 × 7 のグリッド空間において餌を拾うタスクの λ の値を変えて実験 (目標数 · · · 捕食者の数:1, 餌の数:3)

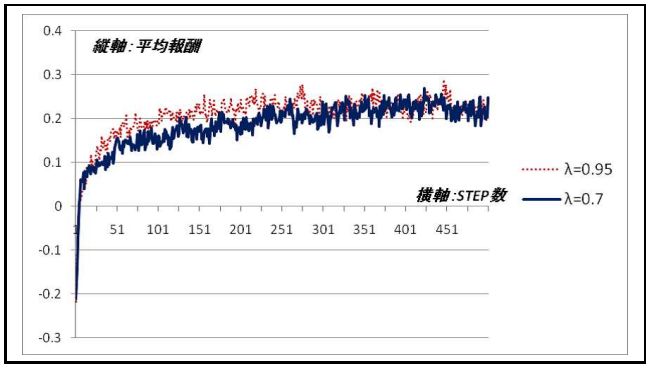

図 6: 7 × 7 のグリッド空間において餌を拾うタスクの λ の値を変えて実験 (目標数 · · · 捕食者の数:1, 餌の数:1)

3.3 考察

3.3.1 減衰係数 λ と学習速度の関係

高速 GM-Sarsa(λ) において，状態行動対のトレースの計算時に使用する減衰係数 λ の値を高くすればする程，学習速度は速くなる．これは，適格度トレースによる Q 値の更新方法から考えると自然な結果と言える．しかし，あまり高すぎると非常に早い段階で学習は収束するものの，多くの収益を集められなくなる (局所解に陥ってしまう)．図 4 は，高速 GM-Sarsa(λ) においてタスク 2~4(餌を拾うタスク) の λ の値を 0.2 にした場合，0.975 にした場合，0.8 にした場合の 3 パターンに分けて先程の実験を行った結果を表しているのだが，この結果からもそのことが分かる．λ の値が高すぎると局所解に陥ってしまう原因として，経験回数の多い状態行動対の Q 値を過大にしてしまう，つまり各 Q 値が正しい期待報酬値を近似できないためであると考えられる．このことから，λ の値は適度に高い値にすべきであると言える．しかし，どのくらいの値が適度なのかは，タスクの数や目標への到達のしやすさに依存する．例えば，7 × 7 のグリッド空間で餌の数が 3 個の場合，餌を集めるタスクの λ は 0.95 よりも 0.7 の方がより適当な値だということが図 5 から分かる．しかし，餌の数が 1 個の場合は，餌を集めるタスクの λ は 0.95 の方がより適当な値だということが図 6 から分かる．

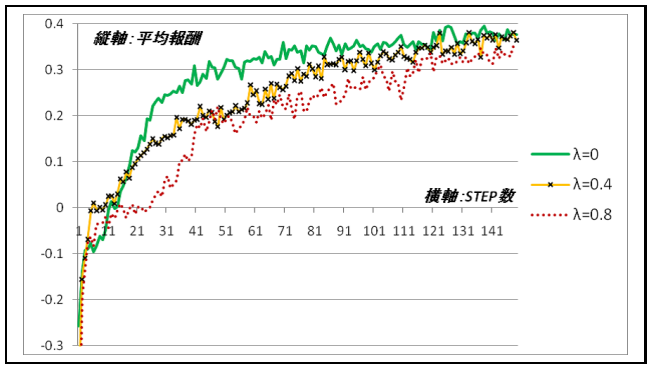

図 7: 捕食者から逃げるタスクの λ の値を変えて実験

さらに，タスクによっては λ の値を低くくした方が学習速度が速くなる場合もある．図 7 の結果は，その典型的な例と言える．図 7 は，高速 GM-Sarsa(λ) においてタスク 1(捕食者から逃げるタスク) の λ の値を 0 にした場合，0.4 にした場合，0.8 にした場合の 3 パターンに分けて先程の実験を行った結果を表している．このような結果が現れる理由は以下の通りである．このタスクの λ の値を上げることによって，このタスク自体の学習速度は上がる．そもそも捕食者に捕まるということは学習の初期段階で多く経験できるので，タスク 1 の学習速度は他のタスクに比べて速い．つまり，このタスクの λ の値を上げると，このタスクの学習は他のタスクよりもずっと速く完了する．しかし，高速 GM-Sarsa(λ) では各タスクにおける Q 値の合計値が最も大きくなる行動を選択するので，タスク 1 の学習のみ完了している場合，各 Q 値が正確な期待報酬値を近似しているタスクがタスク 1 のみなので，必然的にタスク 1 に従った行動を選択することになり，他のタスクにおいて目標に到達するという経験が少なくなってしまう．そのため，他のタスクの学習が遅れてしまい，タスク全体の学習速度も遅くなってしまう．このことから，学習速度を速くするためには，各タスクの λ の値を全タスクがほぼ同時に学習が完了するような値に設定すべきとも考えられる．

3.3.2 非ブートストラップ型学習アルゴリズムとの比較

非ブートストラップ型の学習アルゴリズムは，他の状態行動対の Q 価とは独立に状態行動対の Q 価を更新するため，全ての状態行動対の価値が収束しなくても，よく使われる状態行動対の価値は収束することが多い [7]. そのため，ブートストラップ型に比べて学習速度は速い. 従って，学習の高速化を目指すならば非ブートストラップ型の学習アルゴリズムを用いるべきかもしれない.

そこで，非ブートストラップ型学習アルゴリズムの 1 つである ProfitSharing を用いて各タスクの Q 値を更新する GM-ProfitSharing(以降，GM-PS と呼ぶ) というものを考案した．この GM-PS は各タスクの Q 値の更新以外は GM-Q と全て同じである．各タスクの Q 値の更新方法は次の通りである．タスク T_i の学習器は，タスク T_i において報酬を獲得した時 ($t = episode$) に今まで使用した状態行動対 $[s_{i,t},a_t]$ の Q 値の $Q_i(s_{i,t}, a_t)$ を次の式のように更新する.

$$Q_i(s_{i,t}, a_t) \leftarrow Q_i(s_{i,t}, a_t) + C_{bid}[r_i(t) - Q_i(s_{i,t}, a_t)] \quad (t = 0, \cdots, episode - 1) \tag{6}$$

ここで，$r_i(t)$ はタスク T_i における報酬関数を意味している.

このアルゴリズムを用いて先程の実験を行った．その結果を図 8 に示す．この結果を見て分かるように，確かに学習の収束は速いかもしれないが，多くの収益を獲得することには失敗している．これは，ProfitSharing は多く経験した状態行動対を重視する傾向がありなお且つ，餌集めタスクにおいて学習の初期段階では餌の獲得よりも敵に捕獲されることの方が圧倒的に多いためだと思われる．このことより，複数の目標を持つタスク (特に各目標への到達のしやすさが異なる場合) において，全状態行動対の Q 値を正しい値に近似することは重要だと言える．つまり，このようなタスクを学習する際，ブートストラップ型アルゴリズムは必要不可欠である.

以上より，収益を出来るだけ多く集めながら複数の目標を持つタスクにおける学習の速度をあげるには，非ブートストラップ型学習アルゴリズムを用いるのでなく，ブートストラップ型学習アルゴリズムとその学習速度を上げる適格度トレースを用いるべきだということが言える.

3.3.3 複雑なタスクの学習

複雑なタスク (この場合，目標の種類が多いタスクを指す) における高速 GM-Sarsa(λ) の有用性を検証するため，さらに次のような実験を行った.

実験に用いたタスクは餌集めタスクで，実験条件は実験のセクションで説明したものとほぼ同じである．先程の実験との相違点のみ以下に示す.

- 餌 3 つのうち 1 つを動くものとする．なお，この動く餌を獲物と呼ぶことにする.
- 獲物は常にエージェントから遠ざかる行動を選択し，2STEP に 1 回行動する.

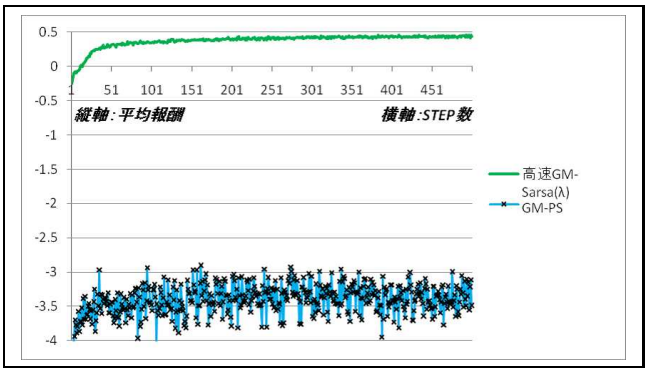

図 8: GM-PS と高速 GM-Sarsa(λ) の比較

- 獲物が 1 回の行動で行えるものは，上下左右のいずれかへ 1 マス移動の 4 種類である.
- 獲物を捕獲する (獲物と同じマスに止まること) と，その獲物は次の STEP には別の座標にランダムに配置される.
- 獲物の捕獲時の報酬は 7，獲物以外の餌の獲得時の報酬は 2，捕食者に捕獲された時の報酬は-10 とする.

さらに，タスクを次のように分割する.

タスク 1 捕食者から逃げるタスク (考慮すべき物 : エージェント自身の位置，捕食者の位置)

タスク 2 獲物を捕獲するタスク (考慮すべき物:エージェント自身の位置，獲物の位置)

タスク 3 餌 1 を拾うタスク (考慮すべき物 : エージェント自身の位置，餌 1 の位置)

タスク 4 餌 2 を拾うタスク (考慮すべき物 : エージェント自身の位置，餌 2 の位置)

この実験を GM-Sarsa(0) と高速 GM-Sarsa(λ) のそれぞれの手法を用いて行った．その結果を図 9 に示す．この結果から，学習速度はどちらの手法も同等であるが，高速 GM-Sarsa(λ) の方が多くの収益を集められるということが分かる．その原因は以下の通りである．獲物を捕獲するタスクの学習よりも先に餌を獲得するタスクの学習が完了してしまう．さらに餌の獲得のタスクの学習が完了していて獲物の捕獲のタスクの学習が完了していない場合，各タスクにおける Q 値の合計値が 1 番大きくなる行動を選択するため，餌の獲得のタスクに従った行動を選択してしまう．そのため，獲物を捕獲するという経験はさらに少なくなってしまい，GM-Sarsa(0) では獲物の捕獲のタスクの全状態行動対を正しい期待報酬値に近似させるのが非常に遅くなり，有限時間では最適解 (収益を最大化させるための行動) を求めることが難しくなる．しかし高速 GM-Sarsa(λ) ならば，獲物の捕獲のタスクの経験数が少なくても，適格度トレースを用いているのでそのタスクの全状態行動対を正しい期待報酬値に素早く近似することができる．そのため，高速 GM-Sarsa(λ) では有限時間で最適解を求めることが可能である.

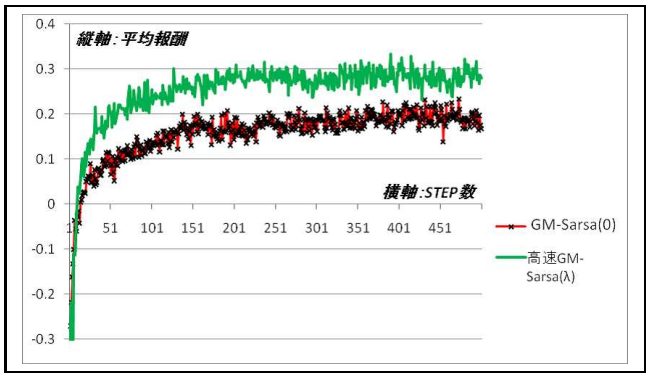

図 9: GM-Sarsa(0) と高速 GM-Sarsa(λ) による複雑なタスクの学習結果

以上より，高速 GM-Sarsa(λ) は学習の高速化だけではなく，複雑なタスクにおいて有限時間内に最適解を求める上でも有効であると言える．

4. おわりに

本論文では，複数の目標を持つタスクにおいて，多くの収益を集めながら行動回数と計算時間の両方の意味で高速に学習することができるアルゴリズム高速 GM-Sarsa(λ) を提案した．その結果，強化学習の現実タスクへの適用の可能性を広げることができたと言える．しかし，強化学習を本格的にロボット等に実装するためには，最低でも以下の 2 点を解決する必要があると思われる．1 つは，POMDP(部分観測マルコフ決定) 環境下での学習を可能にすることである．これは，エージェントが学習するタスクではマルコフ決定過程が成り立つことを前提に学習したが，現実のタスクで MDP が成り立つことは殆んどありえない．つまり現実のタスクでは，今回実験に用いたタスクのように大局的な情報を得ることは不可能で，エージェント自身の視覚等による部分的な観測から得られる局所的な情報を使用して学習しなければならない．このことから，POMDP における学習は重要であると言える．もう 1 つは，連続状態空間における学習を可能にすることである．これは，今回はグリッド空間という離散状態空間における学習を仮定していたが，現実のタスクの状態空間は連続状態空間であることが多いためである．

参考文献

[1] 木村元，宮崎和光，小林重信，“ 強化学習システムの設計指針，” 計測と制御，Vol.38 No.10，pp.1-6，1999.

[2] J.Karlsson，“ Learning to Solve Multiple Goals，” PhD thesis, University of Rochester, 1997.

[3] N.Sprague and D.Ballard，“ Multiple-Goal Reinforcement Learning with Modular Sarsa(0)，” Technical Report 798，University Of Rochester Computer Science Department, 2004.

[4] 森紘一郎，山名早人，“ 強化学習並列化による学習の高速化，” 電子情報通信学会技術研究報告，AI-2003-91，pp.59-64，2004.

[5] R.Sutton and A.Barto，Reinforcement Learning, The MIT Press，1998.

[6] 片山晋，小林重信，“ TD(λ) の対数時間更新算法，” 人工知能学会誌，Vol.14，No.5，pp.879-890，1999.

[7] 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.