日本機械学会論文集(C編)
74巻739号(2008-3)

論文 No. 07-0259

探索エージェントを導入した学習経験を共有するマルチエージェント強化学習システムの提案*

舘山武史*1, 川田誠一*2, 下村芳樹*1

Parallel Reinforcement Learning Systems Using Exploration Agents

Takeshi TATEYAMA*3, Seiichi KAWATA and Yoshiki SHIMOMURA

*3 Faculty of System Design, Tokyo Metropolitan University,
6-6 Asahiga-Oka, Hino-shi, Tokyo, 191-0065 Japan

We propose a new strategy for parallel reinforcement learning ; using this strategy, the optimal value function and policy can be constructed more quickly than by using traditional strategies. We define two types of agents : the exploitation agents and the exploration agents. The exploitation agents select actions mainly for exploitation, and the exploration agents concentrate on exploration using the extended k-certainty exploration method. These agents learn in the same environment in parallel and combine each value function periodically. By using this strategy, the construction of the optimal value function is expected, and the optimal actions can be selected by the exploitation agents quickly. The experimental results of the mobile robot simulation showed the availability of our method.

Key Words: Parallel Reinforcement Learning, Q-Learning, Extended k-certainty Exploration Method, Policy Iteration, Exploration-Exploitation Dilemma, Dyna-Q

はじめに

強化学習法[1]は, 学習エージェントが未知環境と相互作用を行うことにより, 環境に適応した自身の制御法を獲得していく機械学習法であり, ロボットの学習制御などにおける実用化が期待されている. しかし, この手法はタスクによっては学習が収束するまでに膨大な試行が必要であることが重大な問題点として指摘されている. 近年, この問題を解決するため, ある一つのタスクを学習させる際に複数台の強化学習エージェントを同時に学習させ, 経験を共有, または定期的に合成することにより, 短時間で学習を収束させる手法[5][6][7][8][10]が提案されている. これらの手法を用いることにより, 一台のエージェントで単独学習を行うよりも短時間で学習を収束させることが可能となるが, これらの多くの手法では, 学習中は全てのエージェントが共有, 合成した価値関数の大小に基づいた類似した行動戦略をとるため, エージェント間の行動選択の重複による学習効率の悪化や, 探索の不足による局所解での収束などの問題が発生する場合がある. つまり, より良い解を効率良く得るためには, 個々のエージェントがどのような行動戦略をとればよいかについて考察することが重要であるが, これに関する既存研究は十分に行われていないのが現状である. 本論文では, 経験共有を導入したマルチエージェント強化学習システムにおいて, 探索に集中するエージェントと報酬獲得に集中するエージェントを明確に分けて同時学習させることにより, 効率的かつ高い確率で最適解を得ることができる手法を提案する. 本手法では, 探索に集中するエージェントの行動選択法として, PIA 内挿型 k-確実探査法[4]を用いる. 本提案手法を用いることにより, 下記のような利点を実現することが可能となる.

(1) 探索エージェントによる, エージェント間の行動選択の重複が少ない効率的な探索
(2) 報酬獲得エージェントによる, 学習中の定期的な報酬の獲得
(3) 学習が局所解で終了することの防止
(4)Boltzmann 選択や ε -greedy 法[1]のみを用いたシステムよりも, 報酬獲得と探索のバランスをとることが容易 (実数パラメータの細かな調整が不要)
(5) 報酬獲得エージェントのふるまいを観測することにより, 学習の進み具合 (その時点での最適なふるまい) を容易に評価可能

* 原稿受付 2007年3月23日.
*1 正員, 首都大学東京システムデザイン学部(〒191-0065 日野市旭が丘6-6).
*2 正員, 産業技術大学院大学情報アーキテクチャ専攻(〒140-0011 東京都品川区東大井1-10-40).
E-mail : tateyama@sd.tmu.ac.jp

また, 本論文では経験共有の計算コストを削減することを目的として, 複数エージェント間でDyna-Q[11]を実行するアルゴリズムも併せて提案する. 本論文では, 移動ロボットの迷路タスクによって, 上記の提案手法の有効性を実証する.

1. 基礎理論

1·1 Q-learning Q-learning[2]は, 離散MDPs(Markov Decision Processes) 環境[1]での強化学習によく用いられる, 代表的な強化学習アルゴリズムである. 時刻 t で観測された状態 $s_t \in S$ においてエージェントが行動 $a_t \in A$ を実行し, 時刻 $t+1$ で状態 s_{t+1} に推移し, 報酬 r_{t+1} を得たとすると, 状態 s_t の行動価値関数 $Q(s_t, a_t)$ は, 次式を用いて更新される.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)] \quad (1)$$

ここで, $\alpha(0 < \alpha \leq 1)$ は学習率, $0 < \gamma \leq 1$ は割引率である. 行動価値関数とは状態と行動の組に対する評価を見積もる関数であり, それぞれの状態と行動の組の評価値をQ値と呼ぶ. 環境が離散有限MDPs環境であれば, 十分な試行によりQ値が最適値に収束し, 最適政策を獲得できることが保証されている.

1·2 行動選択アルゴリズム Q-learningなどの強化学習法では, 行動選択のアルゴリズムとして ε-greedy法やBoltzmann分布を用いた選択法[1]が良く用いられる. ε-greedy法では, 状態 s_t において確率 $\varepsilon(0 \leq \varepsilon \leq 1)$ でランダムな行動, 確率 $1-\varepsilon$ でQ値が最大となる行動を選択する. また, Boltzmann分布を用いる行動選択法では, 温度定数と呼ばれるパラメータによって調整された, 状態 s において行動 a を選択する確率 $P(s,a)$ を計算する. これらの行動選択法を用いる場合, 学習中の報酬獲得と探索のバランスに影響するパラメータである ε や T を設計者が実験を繰り返し, 試行錯誤で調整する必要がある場合が多い.

1·3 k-確実探査法とPIA内挿型k-確実探査法 k-確実探査法[3]は, 学習中の報酬獲得を考慮せずに環境を効率良く同定することに徹した行動選択手法であり, 行動価値関数を学習していくQ-learningとは異なり, 少ない試行回数で効率良く離散MDPs環境のモデルを構築するための情報を収集することを目的とした手法である. この手法では, 感覚入力に対して実行可能な行動を「ルール」と呼び, それまでの選択回数が k 回以上となっているルールを「k-確実」, そうでないルールを「k-未確実」であると呼ぶ. そして, あるルールを選択すると以後選択できるルールが全て k-確実であるようなルールを「k-確実なループに至るルール」と呼び, そのようなルールを探索中の選択候補から外して無駄な行動を避けることにより, 少ない行動回数で全てのルールを最低 k 回実行しようというものである. k の最大値は設計者が設定し, 全てのルールが k 確実となった時点で, 構築した環境モデルからPolicy Iteration Algorithm(PIA)[1]などによって最適政策を求める.

また, 以上のアルゴリズム中にPIAを内挿したものがPIA内挿型 k-確実探査法[4]である. このアルゴリズムでは, 現状態において k-未確実ルールが存在する場合は k-確実探査法と同様にそのようなルールを優先して選択するが, そのようなルールが存在しない場合は, それまでの探索によって得られた環境情報を利用して部分的に環境モデルを構築し, PIAによって k-未確実ルールが選択できる状態に遷移するための最適政策を導出する. これにより, 探索のランダム性に依存する部分を少なくすることができ, 結果として環境同定の効率化が期待できる. アルゴリズムのさらなる詳細は, 文献[4]を参照されたい.

1·4 学習経験を共有する複数エージェントによる強化学習 環境の状態空間の状態数が膨大である場合, Q-learningなどの強化学習法では学習が収束するまでに膨大な試行が必要となってしまうことが問題である. そこで, 同一環境内で複数の強化学習エージェントを同時, 並列に学習させ, それらの経験を共有, 合成することで学習に要する時間を短縮するアプローチがこれまでにも複数提案されている. Tanは, 政策を共有または定期的に合成することにより, 学習時間が短縮できることをマルチエージェント追跡問題で示している[5]. Kretchmarは, 個々のエージェントの価値関数を定期的に合成し, 学習の収束に要する時間の短縮を実現している[6]. Laurentらは, 並列強化学習 (parallel Q-learning) をマニピュレータのblock-pushing problemに適用し, その有効性を示している[9]. 森らは, 複数のエージェントが共有メモリ上にある共通の価値関数を非同期並列的に更新することにより, 並列計算機によ

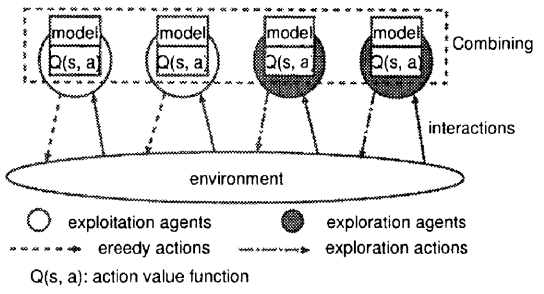

Fig. 1 Parallel Reinforcement Learning System Including Exploitation and Exploration Agents

る計算時間の短縮という立場から高速化を実現している[7][8]. 飯間らは, エージェントを複数用意して学習経験を共有する強化学習法を, 複数エージェントが協調・競合するマルチエージェント強化学習法と区別し, "群強化学習法" と呼ぶことを提案し, エージェント間の情報交換の手法について考察を行っている[10].

2. 探索エージェントを導入した学習経験を共有するマルチエージェント強化学習システムの提案

2·1 対象となる問題設定　本研究では, 複数の強化学習エージェントが学習経験を共有することにより学習の高速化を図る先行研究と同様に, 以下のような問題設定を対象とする.

(1) 同一環境内で複数のエージェントが同時に学習を行う

(2) 学習するべきタスクは全エージェント共通であり, タスクを達成するためにはエージェント間の協調は必要とせず, 一台でも達成可能である

(3) 個々のエージェントは互いに通信が可能である

(4) システム全体の最終目標は, 全エージェント共通の一つの最適タスク実行法を得ることである

また、本研究では強化学習の適用環境を離散 MDPs 環境に限定し, 離散 MDPs 環境を対象とした強化学習アルゴリズムのみを扱う. また, 問題を単純化するため, エージェント間の衝突などの干渉は無視できるという仮定を併せて導入する.

2·2 アルゴリズムの詳細　経験の共有をマルチエージェント強化学習システムに導入することにより, 一台のエージェントで単独学習を行うよりも短時間で学習が収束可能となることが多くの先行研究で示されている (例えば 1.4 に記した文献[5][6][7][8][9][10]). しかし, これらの手法の多くは, 学習中は全てのエージェントが共有, 合成した価値関数の大小に基づいた, 類似した行動戦略をとるため, 行動選択のエージェント間の重複による学習効率の悪化や, 探索の不足による局所解での収束などの問題が発生する場合がある. しかし, より良い解を効率良く得るためには, 個々のエージェントがどのような行動戦略をとればよいかについて考察することが必要である.

そこで本論文では, 経験共有を導入したマルチエージェント強化学習システムにおいて, 探索に集中するエージェントと報酬獲得に集中するエージェントを明確に分けて同時学習させることにより, 効率的かつ高い確率で最適解を得ることができる行動戦略を提案する. 図 1 に, 提案する手法の概念図を示す. 本提案手法では, 学習を行う複数のエージェントを, **報酬獲得エージェント (Exploitation Agents)** と **探索エージェント (Exploration Agents)** の 2 種類に分ける. 報酬獲得エージェントの行動選択アルゴリズムは完全な greedy 選択 (各状態で Q 値が最も高い行動を選択する) か, greedy に近い行動選択 (例えば, ε-greedy 法を用いて ε の値をほぼ 0 に近い値に設定する) であり, その時点で最も高い報酬獲得が期待できる行動を主として選択する. 一方, 探索エージェントは学習中の報酬獲得は考慮せず, 環境全体を網羅的に探索することに集中する. 探索エージェントの行動選択アルゴリズムには, MDPs 環境を効率良く探索するための有効な手法である宮崎らの k-確実探査法[3]を改良した PIA 内挿型 k-確実探査法[4]を用いる. これらの探索手法は, 全ての状態と行動の組 (ルールと呼ぶ) を可能な限り少ない試行回数で k 回以上実行し, 段階的に精度の高い環境の MDPs モデルを構築していくことを目的としたアルゴリズムである. アルゴリズムの詳細は, 各文献[3][4]を参照されたい.

本提案システムの学習のおおまかな流れを以下に示す. まず, 報酬獲得エージェントと探索エージェントは, それぞれの役割に応じた行動選択アルゴリズムに従い, 行動を実行していく. 全エージェントは個々の行動価値関数を持っており, 1 ステップ毎に Q-learning のアルゴリズムに従ってこの関数を更新する. 本提案手法では, 行動選択に行動価値関数を用いない探索エージェントも同様に価値関数を更新するものとする. さらに, 本提案手法では個々のエージェントは環境モデルのデータをルールの実行結果から構築する. ここで, モデルのデータとは, (1) 状態 s で行動 a を実行した回数 n_s^a(ルール実行回数), (2) その結果状態 s' に遷移した回数 $N_{ss'}^a$, および (3) その結果獲得した報酬の総和 $RS_{ss'}^a$ である. そして, 一定時間経過後, 各エージェントの行動価値関数とモデルデータを合成し, 学習経験の合成を行う. まず, 行動価値関数の合成に関しては本研究では以下の式を用いることとする.

$$Q_{new}(s,a) \leftarrow Q_i(s,a), \quad i = \arg_{j \in N} \max n_j(s,a) \quad (2)$$

ここで, Q_{new} は合成後の行動価値関数, Q_i はエージェント i の行動価値関数, $n_j(s,a)$ はエージェント j の Q 値 $Q_j(s,a)$ を更新した回数, N はエージェント総数である. 上記の式は, 更新回数が最も多い値を最も信頼できる値として合成後の行動価値関数に代入することを表している. また, モデルデータの合成は, 単純に全エージェントのモデルデータの値を合計することによってなされる. 合成された行動価値関数とモデルデータは, 全エージェントの個々の行動価値関数とモデルデータと置き換えられ, 次に合成を行うまで個々の関数・データとして行動選択に利用, 更新される.

2·3 Dyna-Qの導入による計算コストの削減　全エージェントの行動価値関数やモデルを合成するには，エージェント数または状態数が増加した場合に多大な計算コストがかかってしまうことが予想される．そこで本研究では，少ない計算コストで定常的にエージェント間の経験共有を実現し，行動価値関数とモデルの合成回数を削減する目的で，Sutton の Dyna-Q[11]を複数エージェント間で利用する手法を併せて提案する．Dyna-Q は，Q-learning によって学習するエージェントが行動価値関数と同時に環境モデルを構築し，単位時間ごとの行動の実行結果に基づいた価値関数の更新とは別に，モデルを用いた価値関数の更新を並行して行うことにより，Q-learning の学習速度を向上させる手法である．この手法は通常は一台のエージェントで実行するためのものであるが，他エージェントのモデルを利用することにより，自身の経験していないルールの Q 値の更新も行うことができるようになる．本提案システムでは，Dyna-Q のアルゴリズムは以下の手順で実行される．

(1) エージェント i が，ランダムに他エージェント j を選択し，通信によって合図を送る
(2) エージェント j が自身の既知状態の中から 1 つの状態 s とその状態で実行経験のある行動 a をランダムに選択する
(3) エージェント j の環境モデルの状態遷移確率と期待報酬値に従い，遷移先状態 s' と獲得報酬 r を算出する．
(4) エージェント j から i に，データ (s, a, s', r) が送信される
(5) エージェント i が受信したデータを用いて Q-learning のアルゴリズムに従って行動価値関数を更新する

以上の 1 サイクルの手順は 2 台のエージェント間だけで行われ，更新する値も一組の状態-行動の価値のみであるため，行動価値関数とモデルの合成と比較して計算コストは小さいと考えられる．よって，上記のアルゴリズムを前節で提案したシステムに導入することにより，行動価値関数とモデルの合成の頻度が抑えられることが期待できる．

3. シミュレーションによる動作確認

3·1 問題設定　シミュレーションの概略図を図 2 に示す．環境は状態数が 9×9 の障害物なし (環境 1) と障害物あり (環境 2) の 2 種の gridworld であり，1 つのマスが 1 つの状態を表している．なお，3.4.3 節と 3.5 節では，環境 1 を 19×19 に拡大した迷路も用いる．エージェントとなる移動ロボットは単位ステップごとに現状態から隣接する上下左右のいずれかの状態に移動し

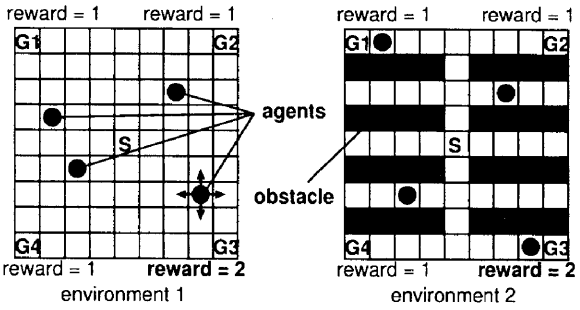

Fig. 2　Mobile Robot Simulation

ようとする行動を選択できる．環境の状態遷移は，非決定性の処理が要求される MDPs の概念に基づき確率的であるとし，0.9 の確率でエージェントが選択した方向の状態に (行動"上"ならば上のマスに)，0.1 の確率でランダムな方向の状態に (行動"上"でも上下左右ランダムのマスに) 移動する．また，外壁に移動しようとする行動を選択すると，衝突して元の状態に戻される．環境内には報酬が獲得できる状態 (G) が 4 つ存在しており，いずれかの状態に到達して報酬を獲得すると，次ステップでスタート地点 (S) に戻される．ロボットの目標は単位時間あたりの獲得報酬値を最大化することである．しかし，環境の状態遷移が非決定的であることに加え，4 つのゴール状態のうち報酬の値が 1 状態だけ 2 倍の値に設定してあるため，学習が局所解 (報酬値 2 を獲得できる状態が発見できない，あるいは最短経路が発見できない) で収束してしまうと，獲得報酬値の合計が低くなってしまう．そのため，十分な探索が必要となる．シミュレーション環境内では，複数のロボットが同時に走行が可能であり，情報を共有するために個々のロボットは他ロボットと相互通信が可能であるとする．また，問題の単純化のために，ロボット同士の衝突，干渉は無いものと仮定し，通信に要する時間も十分短く，無視できると仮定する．

3·2 学習設定　本実験では，行動価値関数の更新には Q-learning を用い，Q-learning のみを用いるシステムのエージェントや本提案手法の報酬獲得エージェントは，行動価値関数の値を用い，ε-greedy 法によって行動選択を行う．また，探索エージェントは PIA 内挿型 k-確実探査法を用いて行動を選択する．全エージェントの共通のパラメータ設定は，Q-learning の学習率 (α=0.1)，割引率 (γ=0.9) であり，ε-greedy 法の ε の値は 0(greedy 選択)，0.05, 0.1, 0.2 それぞれの値で実験を行って結果を比較した．行動価値関数の合成を行う間隔は Dyna-Q を用いた実験を除いて 10step ごとに固定した．実験は各学習設定ごとに 100 回ずつ行い，全ての結果のデータを平均化して比較・考察する．なお，比

較するデータは, Q-learning のみを用いるシステムの場合は任意に選択した一台のエージェントのデータを用い, 提案手法の場合は報酬獲得エージェントのデータを用いた.

3·3　全エージェントが Q-learning のみを用いる場合　まずは, 強化学習の経験共有の先行研究でよく用いられる, 全エージェントが Q-learning を用いて学習し, 同一の行動選択アルゴリズムを用いる手法 (結果グラフ中ではこれらを shared QL と表すこととする) について検証する. 用いた環境は, 図 2 の環境 1 である. 図 3, 4, 5 は, ε-greedy 法の ε の値を 0.05 に固定し, エージェント数を 1, 2, 4 台と変化させた場合の結果の比較を示している. 図 3 はエピソード (スタートから報酬を一回得るまでの区切り) ごとのステップ数の変化であり, エージェントの政策の収束を観察することができる. また, 図 4 は, 獲得した報酬の合計 (代表エージェント 1 台分のみ) の時間推移を示している. これらのグラフから, 経験共有によって学習の高速化と獲得報酬合計の増加が実現されていることがわかる. 次に, これらのシステムが最適な行動価値関数を獲得できているかを調べる. 最適価値関数は, 環境が離散 MDPs 環境で環境構造 (全状態-行動の組の状態遷移確率と報酬期待値) が既知であれば, PIA などの Dynamic Programming を用いて計算可能である[(1)]ため, 実験開始前にあらかじめ図 2 に示した環境の最適価値関数を PIA によって求め、提案システムの評価に用いた. ただし, 学習エージェントは環境構造についてはあらかじめ情報を与えられていないため, 事前に PIA を実行して最適価値関数を求めることは不可能であることに注意されたい. 評価は, エージェントが獲得した価値関数と PIA で求めた評価用の最適価値関数の平均誤差 (各状態の価値の誤差を合計し, 状態総数で割ったもの) を計算し, その時間推移を記録して行う. その結果を図 5 に示す. エージェントの数が増加すると, 価値関数の更新回数も増加するため, 誤差の減少も早くなるが, 100×10^3step 経過時でもまだ誤差が多く残っていることがわかる (環境内の報酬値は 1 または 2 なので, 平均誤差が 0.5 前後という値は小さくないと考えられる). 本研究では, 同じエージェント数でさらに最適な価値関数を短時間で獲得することを目標の一つとしている. また, 図 6, 7 は, エージェント数を 4 台に固定し, ε-greedy 法の ε の値を変化させた場合の結果を示している. 図 6 から, ε=0.05 の場合が最も獲得報酬の合計が高く, greedy の場合が最も低いことがわかる. また, 図 7 からは, ε の値が大きいほど価値関数の誤差が小さくなっていることがわかる. greedy の獲得報酬が低いのは, 探索が不十分であるために最適な解が発見できていないことが原因であり, 価値関数の誤差が小さいにもかかわらず次に ε=0.2 の場合が低いのは, 全エージェントの探索行動の比率が高いためである.

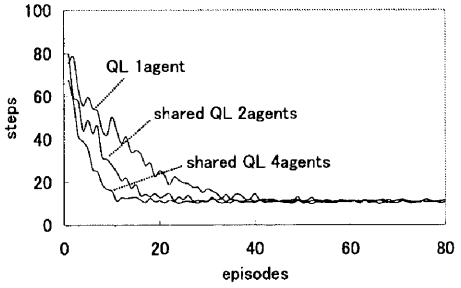

Fig. 3　Episodes-steps graphs of the normal parallel RL systems($\varepsilon = 0.05$)

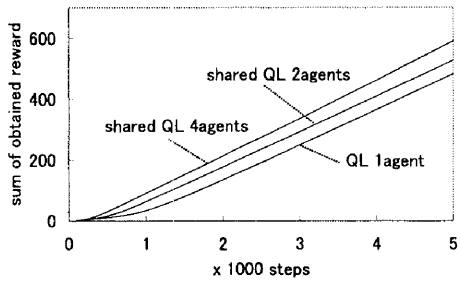

Fig. 4　Sums of obtained reward of the normal parallel RL systems($\varepsilon = 0.05$)

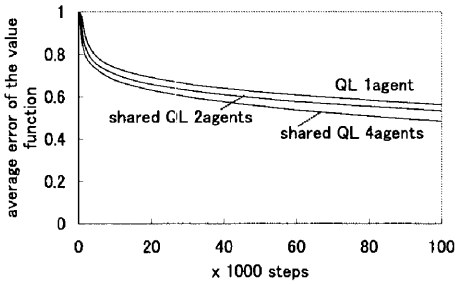

Fig. 5　Errors of value functions of the normal parallel RL Systems($\varepsilon = 0.05$)

3·4　提案手法 1: 探索エージェントを導入した場合

3·4·1　探索エージェントに PIA 内挿型 k-確実探査法を用いた場合　ここでは, エージェント数が 2 台のとき (報酬獲得エージェント 1 台, 探索エージェント 1 台) と 4 台のとき (報酬獲得エージェント 1 台, 探索エージェント 3 台) の場合について図 2 環境 1 を用いて実験を行い, 考察を行う. 本実験では, 提案手法の報酬獲得エージェントの行動選択アルゴリズムは, greedy 選択を用いることとした. 図 8, 9, 10, 11 は, エージェント数が 2 台の場合の結果である. Q-learning のみを用

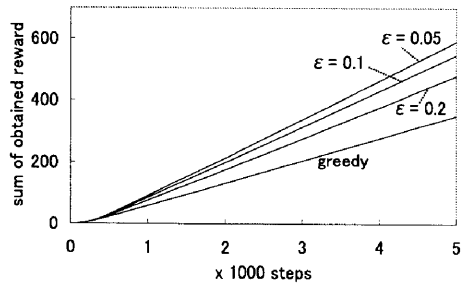

Fig. 6 Sums of obtained reward of the normal parallel RL systems(4agents)

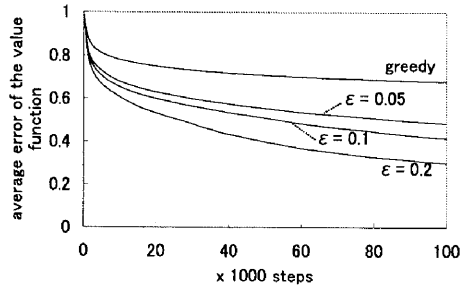

Fig. 7 Errors of value functions of the normal parallel RL systems(4agents)

いた経験共有法では, 行動選択法が greedy の場合, 図11の結果からも分かるように探索が不十分なため価値関数の誤差が十分に小さくなっていない. 加えて, エピソードごとのステップ数も安定せず, 獲得報酬の累積値も低くなってしまっている. このシステムの場合, ε=0.05 として適度に探索行動をとるようにした方が, 良好な結果を得ることが可能であったが, 価値関数の誤差は 100×10^3 ステップ経過時点でも 0.5 程度残っている. これに対し, 探索エージェントを導入した本提案手法では, 報酬獲得エージェントが greedy 行動を取っているにもかかわらず, 図11に示すように報酬獲得エージェントがもつ行動価値関数の誤差が 100×10^3 ステップ経過時点で約 0.013 まで減少している. また, 最適に近い行動価値関数が得られたため, 報酬獲得エージェントは図8の政策収束エピソード付近の拡大図図9に示すように, 報酬獲得までのほぼ最短経路が発見されており (最短経路は8ステップであるが, 結果が8にならないのは 0.1 の確率でランダムな状態遷移が発生するためであると思われる), 図10に示すように獲得報酬の合計値のさらなる増加を実現している. これらの結果は, 本提案手法において探索エージェントが探索に集中することによって (探索に集中しないエージェントと比較して) 効率的に行動価値関数の誤差を減少させ, 行動戦略が異なる2台のエージェントの行動価値関数の合成が, 報酬獲得エージェントの獲得報酬の累積値を増加させるように有効に動作していることを示している.

次に, エージェント数を4とし, 本提案手法において報酬獲得エージェントを1台, 探索エージェントを3とした場合の実験結果の検証を行う. 図12, 13, 14, 15に, 結果のグラフを示す. エージェント数が2台の場合と比較して, 各手法とも学習速度の向上が見られるが, 本提案手法の優位性は変わっておらず, 獲得した行動価値関数の最適性, 獲得報酬の合計値共に本提案手法が最も優れた結果を示した.

3·4·2 探索エージェントの探索アルゴリズムの検証 前節では, 探索エージェントの行動選択アルゴリズムとして PIA 内挿型 k-確実探査法を用いたが, 本節では他の探索アルゴリズムを用いた場合について実験し, 結果を考察する. 実験は, エージェント数を4台, うち3台を探索エージェントとし, 探索アルゴリズムとして PIA 内挿型 k-確実探査法, k-確実探査法, ランダム探索を用いて行った. 1台の報酬獲得エージェントは greedy 選択である. また, 4台が全て ε-greedy 法 (ε = 0(greedy), 0.05) を用いた場合との比較も行った. 図16は, 図2の環境1での価値関数の平均誤差のステップごとの推移を示している. 3種の探索手法の結果を比較すると, 誤差の減少は PIA 内挿型が最も速く, k-確実探査法とランダム探索の結果はあまり大きな差はないことがわかる. 環境1には障害物が存在せず, いわゆる袋小路がない k-確実探査法が有効に動作しにくい環境である[4]ことが原因であると考えられる. 一方, 障害物によって袋小路が数多く存在する環境2は, k-確実探査法が有効に動作しやすい環境である[4]ため, 図17に示すようにランダム探索よりも価値関数の誤差の減少速度が速い. PIA 内挿型は環境構造によらず効率的な探索が可能であり, その結果価値関数の誤差の減少速度も他の手法と比較して極めて速いが, この手法は探索中に PIA を頻繁に実行するため, 1行動選択当たりの計算コストが大きいことが欠点である. このように, 各探索手法は長所と短所を持ち合わせているが, 本提案手法はどの探索アルゴリズムを用いた場合でも, 探索エージェントを導入しない場合と比較して明らかに価値関数の誤差の減少速度が速いことがわかる.

3·4·3 探索エージェントの比率と環境の状態数の増加に関する検証 本節では, 報酬獲得エージェントと探索エージェントの台数の比率がシステムの挙動にどのように影響するかを実験によって考察する. また, 環境の状態総数が増加した場合についても考察する. 実験は, エージェント総数を4台とし, greedy 行動をと

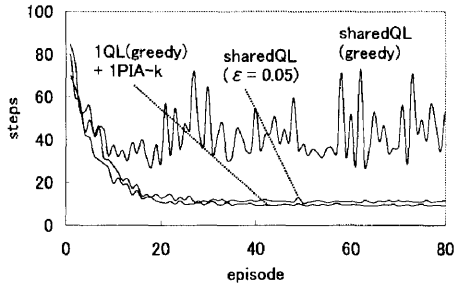

Fig. 8 Episodes-steps graph of the proposed system(1exploitation and 1exploration)

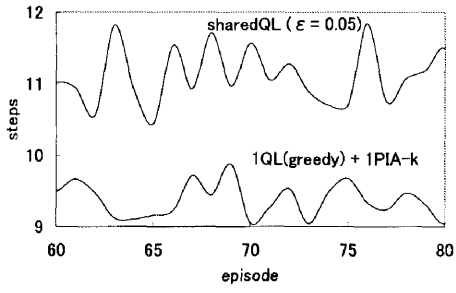

Fig. 9 Closeup of Fig.8(60-80 episodes)

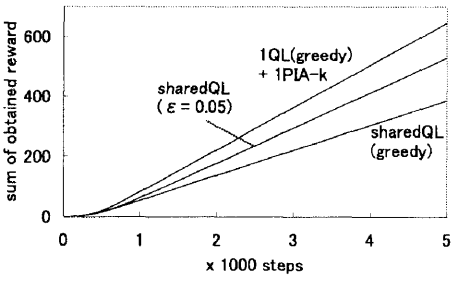

Fig. 10 Sum of obtained reward of the proposed system(1exploitation and 1exploration)

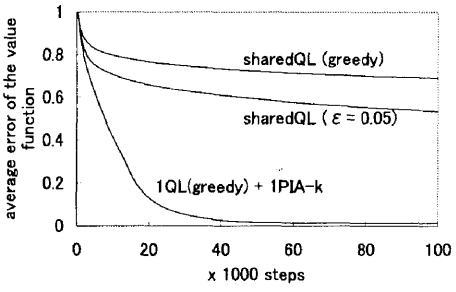

Fig. 11 Errors of value function of the proposed system(1exploitation and 1exploration)

る報酬獲得エージェントと，探索エージェントの比率を 1:3, 2:2, 3:1 と変化させて行った．図 18 は環境 1，図

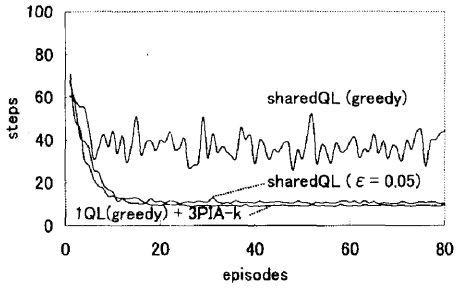

Fig. 12 Episodes-steps graph of the proposed system(1exploitation and 3exploration)

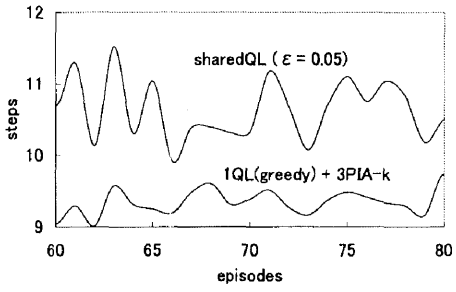

Fig. 13 Closeup of Fig.12(60-80 episodes)

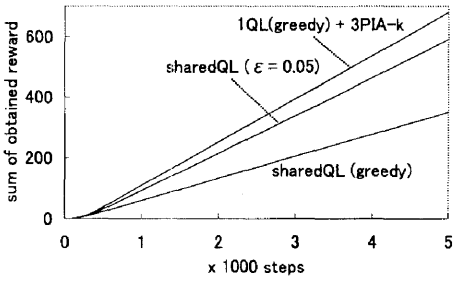

Fig. 14 Sum of obtained reward of the proposed system(1exploitation and 3exploration)

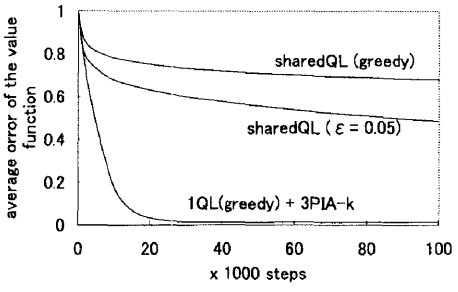

Fig. 15 Error of value functions of the proposed system(1exploitation and 3exploration)

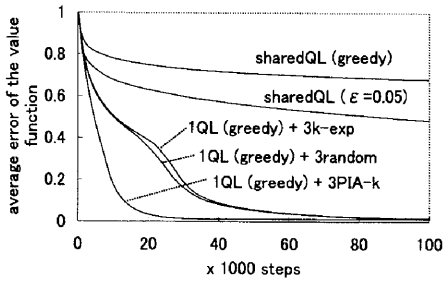

Fig. 16 Average error of the value functions of the proposed systems using three exploration methods(1exploitation and 3explorations, in the 9 × 9 environment which doesn't have obstacles)

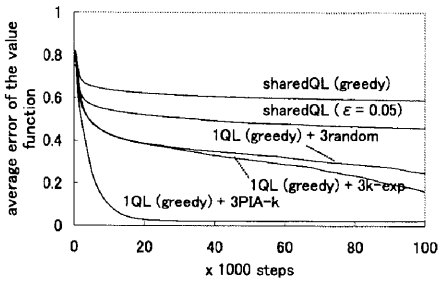

Fig. 17 Average error of the value functions of the proposed systems using three exploration methods(1exploitation and 3explorations, in the 9 × 9 environment which has obstacles)

19 は環境 1 の広さを 19×19 に拡大した環境における価値関数の誤差の推移を示している. これらの結果から, 探索エージェントの数が多いほど合成される価値関数の誤差の減少速度は速まるが, 環境の状態数が多くなるほど, 探索エージェントの数を増加させる効果が大きいことがわかる.

3·5 提案手法 2: 探索エージェント + Shared Dyna-Q を導入した場合 次に, 計算コストがかかる全エージェントの価値関数の合成の頻度を削減する目的で, Dyna-Q を導入したシステムの検証を行う. 実験は, 報酬獲得エージェント 1 台, 探索エージェント 3 台とし, Dyna-Q の実行は各エージェントが 1 ステップごとに 1 回ずつ行うこととした. 用いた環境は環境 1 である. 比較対象は前節で検証した, Dyna-Q を用いずに 10 ステップごとに価値関数の合成を行うシステムであり, Dyna-Q を用いるシステムの価値関数の合成時間間隔を 10, 500, 2000, 5000 ステップと設定した場合と結果を比較した. 図 20, 21 は獲得報酬の合計値の推移グラフであり, それぞれ学習前半部分 (0-2000 ステップ) と後半部分 (十分に学習が収束した 48000-50000 ステップ) を示している. なお, 図中の記号 I_s は, 価値関数の合成を行う時間間隔 [ステップ] である. 学習前半では, Dyna-Q を用いない場合と比較して, I_s の値が等しい場合 (Dyna-Q を用いる手法, 用いない手法共に I_s=10) は早い段階で多くの報酬が得られているが, I_s の値が大きい場合は少ない報酬しか得られていない. これは, Dyna-Q に用いる環境モデルの信頼性が学習初期段階では低いこと, 報酬値の伝搬が十分に行われていない (価値が初期値から変化していない) 状態が多いことなどが原因であると考えられ, 1 ステップごとの Dyna-Q による学習よりも, 全価値関数の合成のほうが学習の高速化の面で効果が高いことを示している. しかし, モデルの信頼性が向上するにつれ, Dyna-Q の効果も上昇し, 最終的には図 21 に示すように I_s = 500 の場合までは比較対象よりも大幅に多くの報酬が得られており, I_s = 2000 の場合は比較対象とほぼ同等の報酬が得られている. I_s = 5000 の場合では, 比較対象よりも獲得報酬の合計が上回ることはなかった. Dyna-Q の効果の

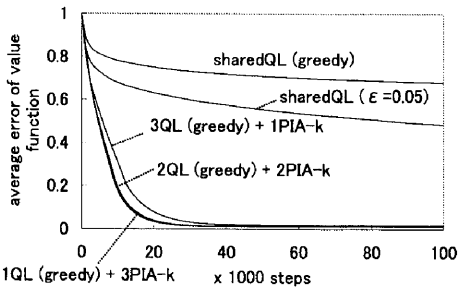

Fig. 18 Errors of the value function(varying proportions of the exploration agents(9 × 9 environment)

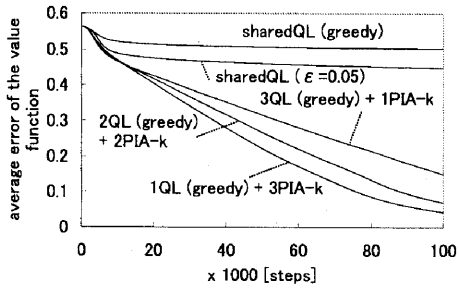

Fig. 19 Errors of the value function(varying proportions of the exploration agents(19 × 19 environment)

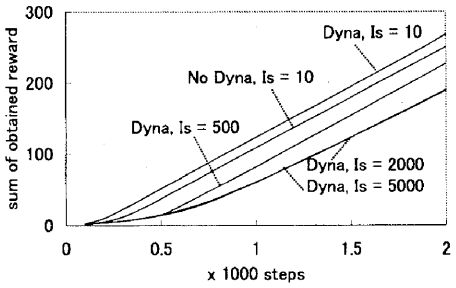

Fig. 20 Sums of obtained reward of the proposed system using Dyna-Q

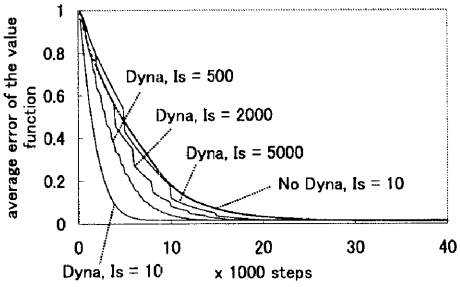

Fig. 22 Errors of value functions of the proposed system using Dyna-Q

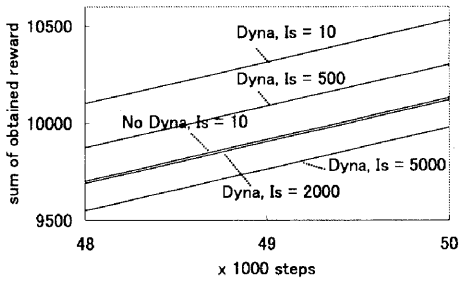

Fig. 21 Closeup of Fig.20(48000-50000 steps)

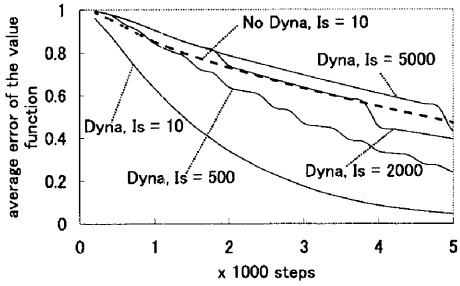

Fig. 23 Closeup of Fig.22(0-5000 steps)

時間的変化は, 次の図 22, 23 に示す価値関数の平均誤差の推移グラフで観察することができる. 前者は学習開始から十分に収束するまでの全体図, 後者は学習初期段階 (0-5000 ステップ) を示したものである. これらの図から, Dyna-Q を用いたシステムが価値関数の合成をきっかけにして段階的に大きく価値関数の精度を向上させていく様子が観察できる.

次に, Dyna-Q の導入による価値関数の合成回数の削減が, どの程度の計算コスト削減の効果があるのかを検証する. ここでは, エージェントを 2 台とし, Dyna-Q を導入したシステムにおいて行動価値関数の合成を行う間隔 I_s を 10 から 5000 まで変化させ, $I_s = 10$ とした Dyna-Q を導入していないシステムとシミュレーション時間を比較することにより, 計算コストの評価を行った. エージェントの行動選択アルゴリズムは, 行動選択における処理内容が一定のランダム探索とした (k-確実探査法や PIA 内挿型 k-確実探査法は探索の進行によって 1 ステップの処理内容が異なり, 計算時間がばらつくことがあるため). 実験は, Pentium4 2.8GHz, メモリ 1GB の計算機で, Vine Linux4.1 上で行った. 実験に使用した環境は 9×9 の環境 1 とそれを 19×19 に拡大した迷路である. 表 1 は, 各設定で 10^5 ステップのシミュレーションに要した時間である (値は 100 回の実験の平均値). この表から, Dyna-Q を導入した本提案システムは, I_s の値を大きく (価値関数の合成回数を少なく) するにつれ, シミュレーション時間が減少していることがわかる. $I_s = 100, 500, 2000, 5000$ のシミュレーション時間は大差ないといえるが, これらの時間を $I_s = 10$ とした Dyna-Q を用いないシステムの結果と比較すると, 約 50%以上減少していることがわかる. 図 20, 21, 22, 23 の結果から, 環境 9×9 の場合は Dyna-Q を用いる提案手法が $I_s = 2000$ 以下であれば $I_s = 10$ とした Dyna-Q を用いない手法と同等, またはそれ以上の報酬の獲得と価値観数のエラーの削減を実現しているといえるため, 前節の提案システムに Dyna-Q を導入し, エージェント間でコンスタントな経験共有を行わせることにより, 計算コストがかかる定期的な価値関数の合成の頻度の削減を実現できたといえる.

4. 実験結果のまとめと考察

まず, 3.4 節の実験により, 本提案手法は従来の経験共有手法と比較して, 価値関数をより小さな誤差でかつ短時間 (少ない行動回数) で収束させ, 結果として報酬獲得エージェントの単位時間あたりの報酬獲得回数を増加させるという結果を得た. これは, 探索エージェントがエージェント間の行動選択の無駄な重複や偏りを削減して効率的な探索を行い, 短時間で価値関数の誤差を減少させることと, 報酬獲得エージェントが定

常的に報酬を獲得し続けることによる効果であり，本提案手法の有効性を示している．この結果から，2.1 節に示した本研究の問題設定においては，本提案手法は短時間で最適に近い解を得るという目的を十分に達成できているといえる．また，本研究では価値関数と環境モデルの合成の頻度を低減し，計算コストを削減することを目的として，Dyna-Q を導入することを提案したが，3.5 節の実験結果はこの有効性を示している．すなわちこの実験結果は，比較的計算コストが少ない Dyna-Q を上述の提案と併用することにより，価値関数と環境モデルの合成の頻度が少ない場合であっても，合成の頻度が多い場合と同等，あるいはそれ以上の学習の高速化を実現できることを示している．また，従来の経験共有を導入した強化学習システムでは，経験共有を行わないシステムと同様に，良質な解を得るための探索行動と学習途中でもある程度報酬を得るための行動のバランスを考慮して設計者が試行実験を繰り返し，エージェントが良好なふるまいを示すように行動選択のパラメータを調整するのが一般的である．しかし，今回の問題設定のように，ほぼ greedy な行動選択法が報酬獲得に有効な環境下では，本提案手法では greedy な行動選択を行う報酬獲得エージェントと複数台の探索エージェントをともに用意するだけで学習結果の最適性と学習途中の報酬獲得の両立が実現できる．よって，本手法は，探索と報酬獲得のバランスを調整する行動選択アルゴリズムのパラメータ調整が容易になるという長所を併せ持つといえる．確率的な行動選択が有効となる環境では，報酬獲得エージェントのパラメータの調整が必要となるが，探索エージェントによって得られた最適価値関数を分析して学習途中で調整するか，異なるパラメータを持つ報酬獲得エージェントを複数同時に学習させるなどの方法が有効であると考えられる．

おわりに

本論文では，経験共有を導入したマルチエージェント強化学習システムにおいて，探索に集中するエージェントと報酬獲得に集中するエージェントを明確に分けて同時学習させることにより，効率的かつ高い確率で最適解を得ることができる手法を提案した．また，経験共有の計算コストを削減することを目的として，エージェント間で Dyna-Q を実行するアルゴリズムの提案を併せて行った．結果として，本提案手法がエージェント間の行動選択の重複が少ない効率的な探索と学習中の定期的な報酬獲得を同時に実現でき，最終的に最適な学習結果が高い確率で得られることを，移動ロボットのシミュレーションによって実証した．今後は，エージェント同士の干渉や，状態数が膨大である環境での対処法など，実環境で本手法を用いる際に無視できなくなる問題について検討していきたい．

Table 1 Computional time(2agents(random walk), 10^5[steps]) *size of the environment

system setting		computational time[sec]	
combination of the value functions	Dyna-Q for parallel RL	$9\times9^*$	$19\times19^*$
don't use	don't use	2.5	11.7
use ($I_s=10$)	don't use	7.4	33.0
use ($I_s=10$)	use	7.7	34.1
use ($I_s=100$)	use	3.5	15.7
use ($I_s=500$)	use	3.1	15.1
use ($I_s=2000$)	use	3.0	12.5
use ($I_s=5000$)	use	3.0	12.6

文献

(1) Sutton, R.S. and Barto, A.G., *Reinforcement Learning An Introduction*, MIT Press(1998).

(2) Watkins, C.J.C.H. and Dayan, P., Technical Note: Q-learning, *Machine Learning* , 8(1992), pp.55-68.

(3) Miyazaki, K, Yamamura, M. and Kobayashi, S., *k*-Certainty Exploration Method: An Action Selector on Reinforcement Learning to Identify the Environment, Journal of Japanese Society for Artificial Intelligence, Vol.10, No.3(1995), pp.124-133 (in Japanese).

(4) Tateyama, T. and Kawata, S., An Efficient Exploration Method Using *k*-Certainty Exploration Method and Dynamic Programming under Markov Decision Processes, Transactions of the Japanese Society for Artificial Intelligence, Vol.16, No.1 B(1995), pp.11-19 (in Japanese).

(5) Tan, M., Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents, Proceedings of the 10th International Conference on Machine Learning(1993), pp.330-337.

(6) Kretchmar, R.M., Parallel Reinforcement Learning, Proceedings of the 6th World Conference on Systemics, Cybernetics, and Informatics, Vol.6(2002), pp.114-118.

(7) Mori, K. and Yamana, H., A Fast Learning Method for Reinforcement Learning on Shared Memory Multiprocessors, Technical Report of IEICE(The Institute of Electronics, Information and Communication Engineers), Vol.2004, No.29(2004), pp.89-94.

(8) Mori, K. and Yamana, H., Parallel Learning Methods of Reinforcement Learning on Shared Memory Multiprocessors, FIT2004(2004), pp.291-292.

(9) Laurent, G. and Piat, E., Parallel Q-Learning for a block-pushing problem, Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems(2001), pp.286-291.

(10) Iima, H. and Kuroe, Y., Swarm Reinforcement Learning Algorithm Based on Exchanging Information among Agents, Transactions of the Society of Instrument and Control Engineers, Vol.42, No.11(2006), pp.1244-1251.

(11) Sutton, R.S., Integrated Archtectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming, Proceedings of the Seventh International Conference on Machine Learning(1990), pp.216-224.